D2.1 ICOS ecosystem: Technologies, requirements and state of the art

Document Identification			
Status	Final	**Due Date**	28/02/2023
Version	1.0	**Submission Date**	10/03/2023

Related WP	WP2	Document Reference	D2.1
Related Deliverable(s)	N/A	Dissemination Level (*)	PU
Lead Participant	ATOS	Lead Author	Francesco D'Andria
Contributors	ATOS, NCSRD, PSNC, L-PIT, SUITE5, XLAB, ENG, UPC, SSEA ZETTA, RHT, BSC, CeADAR, SIXSQ, TUBS, NKUA, CRF	Reviewers	Simone Ferlin-Reiter (RHT)
			Nabil Abdennadher (SixsQ)

Keywords:
Cloud-Edge-IoT Continuum Management, Cognitive Cloud, Artificial Intelligence, Data Management over the continuum, Trustworthy

This document is issued within the frame and for the purpose of the ICOS project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101070177. 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 ICOS Consortium. The content of all or parts of this document can be used and distributed provided that the ICOS project and the document are properly referenced.
Each ICOS Partner may use this document in conformity with the ICOS Consortium Grant Agreement provisions.

(*) Dissemination level: **(PU)** Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page). **(SEN)** Sensitive, limited under the conditions of the Grant Agreement. **(Classified EU-R)** EU RESTRICTED under the Commission Decision No2015/444. **(Classified EU-C)** EU CONFIDENTIAL under the Commission Decision No2015/444. **(Classified EU-S)** EU SECRET under the Commission Decision No2015/444.

Document Information

List of Contributors	
Name	Partner
Alex Volkov	ATOS
Marc Fabián	ATOS
Francesco D’Andria	ATOS
George Xilouris	NCSRD
Nikos Dimitriou	NCSRD
Marcin Plociennik	PSNC
Lukasz Lowinski	L-PIT
Iman Esfandiyar	L-PIT
Konstantinos Latanis	SUITE5
Hrvoje Ratkajec	XLAB
Nejc Bat	XLAB
Aleš Černivec	XLAB
Gabriele Giammatteo	ENG
Jordi Garcia	UPC
Xavi Masip-Bruin	UPC
Sergi Sánchez-López	UPC
Rosalia Davi	SSEA
Ivan Paez	ZSCALE
Sreeja Nair	ZSCALE
Ignasi Garcia-Milà Vidal	WSE
Izabela Zrazinska	WSE
Jose Castillo Lema	RHT
Francesc Lordan	BSC
Anna Queralt	BSC
Gabriel Puigdemunt	BSC
Ricardo Simon Carbajo	CeADAR
Andrés L. Suárez-Cetrulo	CeADAR
Sebastián Cajas Ordóñez	CeADAR
Jaydeep Samanta	CeADAR
Konstantin Skaburskas	SIXSQ
Lionel Schaub	SIXSQ
Khaled Basbous	SIXSQ
Nabil Abdennadher	SIXSQ

List of Contributors	
Name	Partner
Marc Michalke	TUBS
Admela Jukan	TUBS
Francisco Carpio	TUBS
Fin Gentzen	TUBS
Anastasios Giannopoulos	NKUA
Konstantinos Skianis	NKUA
Panagiotis Gkonis	NKUA
Panagiotis Trakadas	NKUA
Marina Giordanino	CRF

Document History			
Version	Date	Change editors	Changes
0.1	04/10/2022	ATOS	First Draft
0.2	19/1//2023	NCSRD	Integrated document, first version
0.3	26/1//2023	ATOS	Second round of UC contributions integrated, Acronyms added, and Introduction added.
0.4	02/02/2023	NCSRD, UPC	Integration of User Stories
0.5	09/02/2023	ATOS, NCSRD	Third round of UC contributions integrated, Executive Summary added, and Conclusions added
0.6	16/02/2023	ATOS, NCSRD,	ICOS Requirements refinement and State of the Art
0.7	23/02/2023	ATOS, NCSRD, RHT, SIXSQ	Feedback from reviews integrated, final request for inputs from UC leaders pending
0.8	24/02/2023	ATOS	Small format changes to fulfill quality guidelines
1.0	10/03/2023	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Francesco D'Andria (ATOS)	10/03/2023
Quality manager	Carmen San Román	10/03/2023
Project Coordinator	Francesco D'Andria (ATOS)	10/03/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AFO	Adaptive Federated Optimization
AGIX	SingularityNET
AI	Artificial Intelligence
AIaaS	Artificial Intelligence as a Service
AODV	On-Demand Distance Vector Routing Protocol
AORP	Agriculture Operational Robotic Platform
API	Application Programming Interface
ARM	Advanced RISC Machine
AuthN	Authentication
AuthZ	Authorization
AWS	Amazon Web Services
BATMAN-adv	Better Approach to Mobile Ad-hoc Networks – advanced
BSD	Berkeley Software Distribution
CA	Certification Authority
CaaS	Container as a Service
CAN	Controller Area Network bus
CC	Cloud Continuum
CFG	Control Flow Graph
CI/CD	Continuous Integration / Continuous Deployment
CIDR	Classless Inter-Domain Routing
CLI	Command Line Interface
CO2	Carbon Dioxide
CP	Cloud Provider
CPU	Central Processing Unit
CRD	Custom Resource Definition
DAG	Directed Acyclic Graph
DC	Data Center
DDS	Data Distribution Service
DNN	Deep Neural Network
DNS	Domain Name System
DoA	Description of Action
DSO	Distribution Supplier Officer

Abbreviation / acronym	Description
DSR	Dynamic Source Routing protocol
DVC	Data Version Control
DX.Y	Deliverable X.Y
ECIP	Edge Computing Infrastructure Provider
ECS	Amazon Elastic Container Service
ECU	Engine Control Unit
ELSA	Edge Lightweight Searchable Attribute-based encryption system
EMCS	Energy-Efficient Makespan Cost-Aware Scheduling
EMDS	Energy Management and Decision Support system
ESBN	Electricity Supply Board Networks
EU	European Union
EV	Electric Vehicle
FaaS	Function as a Service
FCOS	Fedora CoreOS
FedAvg	Federated Averaging algorithm
FedEL	Federated Evolutionary Learning
FedGNN	Federated Graph Neural Network
FedPer	Federated Learning with Personalization Layers
FedProx	Federated Optimization in Heterogeneous Networks
FedSage+	Subgraph Federated Learning with Missing Neighbor Generation
FedTL	Federated Transfer Learning
FGC	Ferrocarrils Generalitat Catalunya
FL	Federated learning
GB	Giga Byte
GDPR	General Data Protection Regulation
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPT-2	Generative Pre-trained Transformer 2
GPU	Graphical Processing Unit
GUI	Graphical User Interface
GW	Gateway
HPC	High Performance Computing
HPE	Hewlett Packard Enterprise
HSV	Hue Saturation Value
HTTP	Hypertext Transfer Protocol
HW	Hardware

Abbreviation / acronym	Description
IaaS	Infrastructure as a Service
IaC	Infrastructure as a Code
IAIMM	In-car Advanced Infotainment and Multimedia Management system
ICOS	Towards a functional continuum operating system
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IL	Intelligence Layer
ILP	Integer Linear Programming
IMU	Inertial Measurement Unit
IOS	iPhone Operating System
IoT	Internet of Things
IoTP	Internet of Things Provider
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LAN	Local Area Network
LoRa	Long Range
LSABE-MA	Lightweight Searchable ABE with multi-authority
MADRL	Multi-Agent Deep Reinforcement Learning
MCC	Mobile Cloud Computing
MFCC	Mel-Frequency Cepstral Coefficients
ML	Machine Learning
MLMD	Machine Learning Metadata
MLOps	Machine Learning Operation
MNIST	Modified National Institute of Standards and Technology database
MQTT	MQ Telemetry Transport
MW	Mega Watt
NBI	North Bound Interface
NDN	Named Data Networking
NFV	Network Function Virtualization
NLP	Natural Language Processing
NN	Neural Network
NNI	Neural Network Intelligence
NP	Network Provider
NP-hard	Nondeterministic Polynomial time

Abbreviation / acronym	Description
OBS	One Big Switch
OCI	Open Container Initiative
OCM	Open Cluster Management
OLM	Open Learner Models
OLSR	Optimized Link State Routing protocol
OS	Operating System
OSI	Open Systems Interconnection model
OT	Operation Technology
OTE	Optimal Trustworthy EdgeAI
P2P	Peer to Peer
PDP	Programmable Data Plane
PLONK	PLONK is a cloud native stack for application developers: Prometheus; Linux; OpenFaaS; NATS; Kubernetes
PoI	Point of Interest
PoP	Point of Presence
PV	PhotoVoltaics
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RDD	Resilient Distributed Dataset
REST	Representational State Transfer
RGB	Red Green Blue
RL	Reinforcement Learning
ROS2	Robot Operating System 2
RSAM	Railway Structural Alert Monitoring system
RTK	Real-Time Kinematic positioning
SBI	South Bound Interface
SDK	Software Development Kit
SDN	Software Defined Networking
SFC	Service Function Chaining
SFV	Sensor Function Virtualization-based
SGX	Intel Software Guard Extensions
SLA	Service Level Agreement
SLO	Service Level Objective
SNR	Signal To Noise Ratio
SplitFL	Split Federated Learning

Abbreviation / acronym	Description
SQL	Structured Query Language
SSOT	Single Source of Truth
StaSA	Service–Time-Aware Scheduling Algorithm
SW	Software
TEE	Trusted Execution Environment
TPU	Tensor Processing Unit
TSO	Transmission Supplier Officer
UC.X	Use Case X
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
US.X	User Story X
VM	Virtual Machine
VPN	Virtual Private Network
WPX	Work Package X
XR	Extended Reality
YAML	YAML Ain't Markup Language

Executive Summary

The ICOS project aims to cover the range of challenges that arise when addressing the Cloud-Edge-IoT continuum paradigm, proposing an approach that embeds a well-defined set of functionalities, culminating in the definition of an IoT2cloud operating system.

ICOS's mission is to design, develop and validate a meta operating system for the Cloud-Edge-IoT continuum, by addressing the following challenges:

- devices volatility and heterogeneity, continuum infrastructure virtualization and diverse network connectivity;
- optimized and scalable service execution and performance, as well as resources consumptions, including power consumption;
- guaranteed trust, security and privacy, and;
- reduction of integration costs and effective mitigation of cloud provider lock-in effects, in a data-driven system built upon the principles of openness, adaptability, data sharing and a future edge market scenario for services and data.

Deliverable D2.1 - ICOS ecosystem: Technologies, Requirements and State of the Art, hereafter referred to simply as D2.1 first introduces the ICOS motivation and challenges. It then defines the ICOS concepts and artifacts. These definitions are tentative and serve to support the Functional and Non-Functional requirements elicitation activity.

The next step was to identify the stakeholders and their roles in the ICOS ecosystem. Those Stakeholders are involved in a series (eight) of User Stories that explain the method by which the ICOS will manage and control Cloud Continuum as well as the key drivers and the incentive that set the context in which solutions and services of the platform should be offered.

The Functional and Non-Functional requirements of the project are also defined in this document. Such activity has been carried out from five different perspectives, which are considered to be the main themes of the project, in order to produce the functional requirements:

- Continuum Creation:
- Continuum Management:
- Data Resiliency and Transformation:
- Smart Security and Trust:
- Operability and Serviceability (GUI/CLI)

Finally, the document discusses the state of the art of services, applications and processes, as well as commercial and open-source projects related to ICOS approaches and future achievements.

1 Introduction

1.1 Purpose of the document

This deliverable presents the outcomes of the work carried out by the activities T2.1 – Ecosystem identification: Baseline technologies; T2.2 – Compute continuum requirements definition and T2.3 – AI, data management and trust/security requirements.

This document purpose is twofold, on one hand to discuss the Use Cases and User Stories that will define the ICOS development in terms of initial requirement elicitation and on the other hand to survey the relevant technologies and frameworks relevant for the implementation of ICOS and satisfaction of the elicited requirements.

Initially this document provides for some initial definitions of terms that are relevant to ICOS and are used throughout the document. Their final definition is expected to be presented at the D2.2. Deliverable on ICOS architecture. Following the definitions, the ICOS ecosystem and its stakeholders is presented.

With the purpose of eliciting requirements, the document discusses system User Stories on the operation of ICOS that detail workflows related to the cloud continuum creation, deployment of a service, service monitoring and optimisation, and data processing and ICOS control. Following the User Stories which are more technical but without revealing too much information on the architectural components, the document introduces the anticipated Use Case that will be used to evaluate the implementation of ICOS highlighting relevant requirements to be analysed. The Use Cases information shared in this document is high level, as at the current stage their processing will be vital for the design and specification of the ICOS architecture, and a detailed technical analysis will not be relevant. The evaluation of ICOS will be carried out through four use cases (UC1: Agriculture Operational Robotic Platform; UC2: Railway Structural Alert Monitoring system; UC3: In-car Advanced Infotainment and Multimedia Management system; UC4: Energy Management and Decision Support system) aimed at confirming the framework's capabilities at the cloud, edge and IoT levels and according to specific constraints...

ICOS Requirements Elicitation is the result of analysis and a series of activities carried out within the different actors and stakeholders of the project's value chain. To provide a set of requirements, User Stories and ICOS Use Cases were studied and the results from mentioned analysis are detailed in a set of tables. This output is produced by following the MoSCoW method to reduce the requirements elicitation extension.

Finalizing the contributions of this deliverable, survey scientific, technology and business trends in the area of Cloud, Edge and IoT computing that are relevant to the ICOS project in the field of the service and infrastructure management, data processing and management, related research project is presented. This work will give a generic overview of all technological trends of which awareness is necessary for the project, and for the proposed area of research at large.

This is the first version of the deliverable (v1), aligned to iteration 1 (IT-1) of the project. A second version (M20), aligned to iteration 2 (IT-2), is due to be delivered within the second project period.

1.2 Structure of the document

The deliverable is structured in eleven main sections:

- Section 2 presents the ICOS motivations and challenges.
- Section 3 provides a preliminary definition on ICOS concepts and artifacts that are discussed in the following sections.
- Section 4 identifies ICOS ecosystem stakeholders and the roles.
- Section 5 introduces eight User Stories which explains the method by which the ICOS manages and controls Cloud Continuum.
- Section 6 provides a detailed description for each of the ICOS Use Cases.
- Section 7 presents the ICOS Requirements Elicitation as the result of analysis and a set of activities implemented within the different actors that constitute the value chain of the ICOS project.
- Section 8 summarizes the relevant technologies, toolkits and frameworks considered for the design and development of the ICOS ecosystem
- Section 9 introduces Continuum data processing and Management tools and services
- Section 10 presents ICOS related state of the art on research projects
- Section 11 concludes the deliverable.

2 ICOS Motivation and Challenges

2.1 Motivation

The unstoppable proliferation of novel computing and sensing device technologies, together with the ever-growing demand for complex data-intensive applications are driving the next wave of transformation in computing systems. Now, data-intensive applications can be executed not only in large cloud infrastructures but can also be offloaded distributedly among an enormous spectrum of heterogeneous computing devices spread around the globe, leveraging the capabilities of each individual device according to the specific application data and computation requirements. The resulting paradigm shift in computing leads to new forms of system architectures and organization, associating seamlessly a myriad of cloud, edge and IoT resources into a single computing system, to form a continuum.

A continuum, today also referred to as cloud continuum, IoT continuum, edge-to-cloud or fog-to-cloud, is expected to provide the means for data processing both at the edge and at the cloud, leveraging the massively distributed nature of the continuum to exploit locality and optimize resources utilization. Effective management of the continuum is a daunting challenge which includes not only device discovery, topology tracking, services orchestration, resources allocation, and data sharing and optimization, but also monitoring and inferring critical information for post-mortem and offline analysis while guaranteeing security and preserving privacy.

Despite the intensive research activities and clear industrial trends in this field, performance of resource usage and efficiency in locality exploitation, as well as contextual intelligence of the continuum are far from being achieved. This is not only due to the continuum being intrinsically heterogeneous, dynamic, volatile, highly distributed, and increasingly cognitive, but also due to the emerging request to be open, transparent and collaborative.

We envision a holistic approach towards the solutioning of this technology trend in future systems, by architecting, designing and implementing the continuum as open, extensible, secure, easy-to-use, adaptable, data-driven, AI-powered as well as highly performant and technology agnostic, managed through a meta OS, i.e., the IoT-to-Cloud Operating System (ICOS).

2.2 Challenges

Conceiving a holistic and comprehensive system of this magnitude built upon the principles of openness, adaptability and data sharing is a daunting task which should address the complexity of devices heterogeneity, mobility and volatility, continuum infrastructure management and topology tracking (including network connectivity and devices dynamic availability), optimized and scalable service execution and performance (according to different metrics, such as response time, energy consumption, etc.), resources allocation, virtualization and clustering, guaranteed trust, security and privacy, as well as reduction of integration costs and effective mitigation of cloud provider lock-in effects.

Specifically, ICOS has been conceived to face the following main challenges:

Challenge 1: Enabling technology agnostic operation in a heterogeneous continuum infrastructure. Unlike centrally managed clouds, massively heterogeneous systems in the continuum (including IoT devices, edge devices and cloud infrastructures) are significantly more complex to manage. Furthermore, distributed data management raises an additional level of complexity by classifying data infrastructures, collecting vast and diverse data volumes, providing transparent data access methods, optimizing the internal dataflow, and effectively preserving data collections. ICOS should be able to provide a transparent and AI-powered operational system to leverage heterogeneous infrastructure in the continuum. Adding new devices and technology increases this challenge as shown in the efforts to leverage the work within the open cluster management upstream project. What is needed in ICOS is the same level of technology-agnosticism in the edge as we today have in the cloud.

Challenge 2: Facilitating an on-demand ad-hoc and AI-assisted development of the continuum infrastructure. The continuum needs to be efficiently managed to optimally meet the application demands during service execution. This includes not only identifying the best set of nodes to execute the application, bringing computation closer to where data is produced, cleaning, formatting and pre-processing data to optimize execution both for real-time processing and non-real time data analytics, but also monitoring the service execution to detect eventual failures and applying the appropriate fault tolerance mechanisms, as well as monitoring the continuum to detect the availability of new resources which could improve the performance and, in this case, apply the corresponding migration mechanisms. Furthermore, an AI-powered system like ICOS should be able to implement not only reactive optimization policies but also sophisticated decision taking mechanisms based on system activities forecasting.

Challenge 3: Considering reliability and trust-by-design. Reliability of the continuum infrastructure is not granted. Due to the distributed and dynamic nature of the continuum, with plenty of devices from different owners and provenance, the application of reliability and trustiness becomes fundamentally challenging. Secure mechanisms to access the distributed nodes, data privacy preservation, and providing open and transparent operation is fundamental to enhance trustiness. ICOS should be conceived trust-by-design, considering the appropriate security mechanisms from the system origin, and fostering the adoption of the AI-powered layer to develop innovative and sophisticated solutions to maximize reliability and trustworthiness.

Challenge 4: Creating an open platform facilitating resources, models, data and services sharing, promoting EU innovation and new business models in the continuum arena. The continuum is a broad and diverse space where there are no one-size-fits-all solutions. Although several standards and open-source projects and foundations focus on this, the envisioned continuum should focus on promoting openness, interoperability, services and data sharing, as well as providing a set of optimized AI and business specific models to efficiently leverage the continuum infrastructure. Contributing to an open ecosystem favours interoperability with existing and emerging frameworks, towards an innovative and collaborative European edge market scenario.

3 ICOS Concepts definitions

The following table provides preliminary definitions on ICOS concepts and artifacts that are discussed in the following sections. These definitions are tentative and serve the purpose of the requirements elicitation. The final definitions are anticipated in the forthcoming deliverable on the ICOS architecture design and specification.

Table 1: Definitions on ICOS concepts and artifacts

Domain	An infrastructure managed and controlled under the provisions of a single administration entity. Maybe further detailed depending on the complexity of the infrastructure topology , the underlying technologies and the NBI provided for third party management/orchestration.
Cloud Continuum (CC)	A group of resources (CPU, memory, network, storage, intelligence) managed seamlessly end-to-end that may span across different administrative/technology domains in multi-operator and multi-tenant settings.
ICOS Agent	The basic ICOS software component that needs to be installed on each node (infrastructure node) capable of executing external software and/or providing data and metadata.
ICOS Controller	The ICOS component/software that is responsible for peering with ICOS Agents for providing control and lifecycle operations.
ICOS Application Lifecycle Controller	Application Lifecycle Controller that manages a pool of resources across CC and is distributed across the ICOS controllers employing E/W interfaces. The Application Lifecycle Controller has the authority to control the lifecycle of the deployed Application/Service over the associated resources (i.e. the CC).
ICOS Element	Is the unitary resource available on a node (e.g. Computing, Storage, Network, Intelligence etc.). The basic, unitary ICOS Element is defined as the minimum set of resources defined as CPU/RAM/Storage (dynamically allocated by the native infrastructure management administration) encapsulated within HW or SW entities such as (VM, container or HW node).
ICOS Node	Any resource physical or virtual that is running either an ICOS Controller or Agent.
Restricted Devices	IoT device or infrastructure element (that can be resource-restricted) controlled through API provided by ICOS Agent on ICOS Nodes. The IoT device or infrastructure element could also be directly accessed via its control interface (e.g., API) i.e., not necessarily through the ICOS Node.
ICOS Instance	A subset of the CC participating in the execution of a multi-component Application of a certain topology, resource requirements and constraints."
ICOS Shell	A client distribution that is used to access the ICOS. The ICOS Shell runs externally to the ICOS Instance.

ICOS Networking
The ICOS will exploit available APIs and build logical connections as required for the realization of ICOS node interconnection or ICOS Instance.
ICOS Application Template
It is provided by ICOS Shell. It is the ICOS Application descriptor containing metadata and allows the definition of application components and deployment within the Cloud Continuum

4 ICOS Ecosystem

The ICOS ecosystem is illustrated in Fig. 2. The identified stakeholders and the roles in the ICOS ecosystem are i) the End User; ii) the Application Developer; iii) the Application Integrator and iv) the infrastructure providers. Further we describe the stakeholders of the ICOS ecosystem.

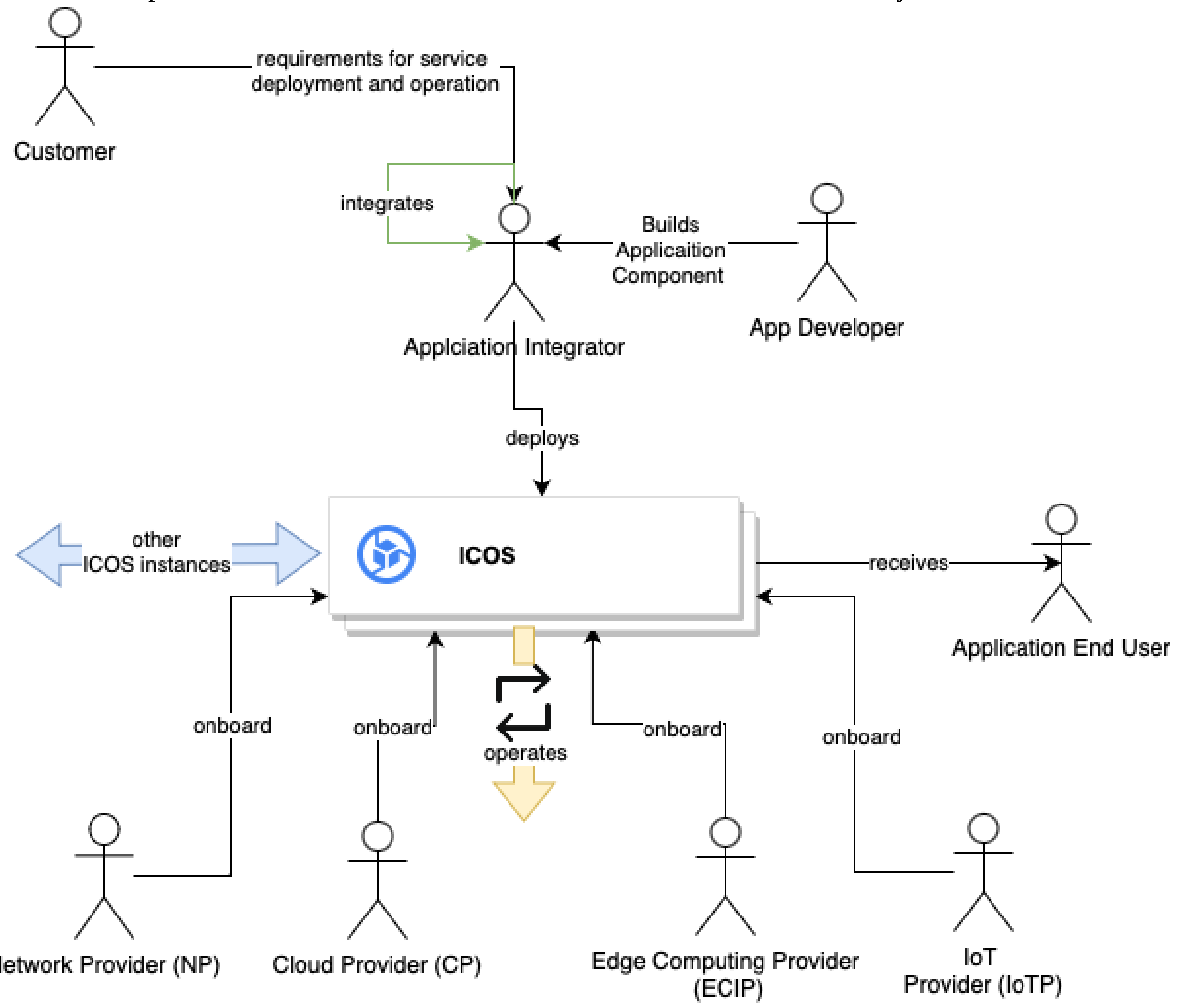


Figure 1: ICOS roles and interactions

Infrastructure Providers

- Network Provider (NP): The NP is providing the network and connectivity resources that allows the internetworking of Cloud with Near Edge and Far Edge locations as well as the provisioning of the required resources supporting GWs and remote devices connectivity. Within the cloud continuum there can be multiple NPs depending on the footprint of the infrastructure and the administrative domains.
- Cloud Provider (CP): The CP is provisioning the cloud resources responsible for hosting the application components. Commonly, the CP operates on large cloud infrastructures (e.g., Hyper scalars), providing points of presence (PoPs) of local interest (i.e., exchange nodes or local DCs in preferred locations) for allowing fast connectivity, low latency, load balancing and close to the devices resource availability. In ICOS the CP is expected to provide infrastructure nodes that will host ICOS in order to deploy ICOS Nodes (controllers or agents)

- Edge Computing Platform Provider (ECPP): Similarly, the ECIP is providing cloud resources at the edge (near or far) of the infrastructure, capable of hosting less resource demanding application's components coupled with specific HW acceleration capabilities suitable for AI/ML workloads. It is assumed that the ECIP infrastructure topology allows for reaching large and/or dense IoT deployments. Similarly, to the CP, the ECIP will allow the deployment of ICOS on its infrastructure nodes in order to extend ICOS to the edge.
- IoT Provider (IoTP): IoT provider is the actor providing the IoT infrastructure that is being deployed across the continuum. This infrastructure may include devices that allow deployment of ICOS controllers or/and agents. Moreover, in the case that the capability to deploy ICOS is restricted either due to processing resources limitations or because of access to the HW device OS. In the latter case, offered APIs are exploited. The IoTP through ICOS is gaining the ability to open the infrastructure to multiple vertical applications.

Application Developer/Application Integrator

The Application Developer (AppDev) and the Application Integrator (AppInt) can be seen as distinguishing roles played by the same actor or different, depending on the complexity of the ecosystem. The first one, is developing Application components, enhancing functionality and operation. The latter one, is integrating Application components that may arrive even from different developers, so that a full-blown Application is created and modelled/discrined in a compatible to ICOS model. In this context the AppInt is experienced with the ICOS data model, descriptor, and operation specificities. Consequently, the AppDev depends on the Application Integrator to formulate the Application descriptors in a way that is comprehensible by ICOS in order to be deployed over an ICOS instance. However, for the current project stage, the AppDev and AppInt are considered to be played by the same actor, that takes into account both the development of the application components as well as integrating everything according to ICOS specifications.

Customer (ICOS User)

The Customer is the actor that uses ICOS to deploy his/her applications over suitable infrastructure, exploiting the ICOS management and optimization capabilities to deploy applications fuelling novel business opportunities.

Application End-User

The End-User of the deployed application provided by the ICOS customer.

Cloud Continuum Provider (CCP)

The CCP is a role that can be undertaken by any Infrastructure Provider that is hosting one or more ICOS controllers in its infrastructure. Hosting the ICOS Controller, allows him to accept requests for application deployment from Customers.

5 ICOS User Stories

This section discusses several basic ICOS system user stories that will allow better understanding of the ICOS operation as well as elicit functional requirements that will be discussed in the following sections. The User Stories presented in this section, provide a high-level overview of the main interactions and workflows anticipated in ICOS and will be further refined in D2.2 Deliverable on ICOS architecture. The section also is complementary to the following section on ICOS Use Cases, attempting to elicit requirements that may have been missed by the Use Cases specified by the project.

5.1 User Story 1 [US.1]: Cloud Continuum Realization

This User Story explains the method by which the ICOS managed and controlled Cloud Continuum is realized. According to the definitions of Section 2, whenever an ICOS Node is bootstrapped, that node immediately becomes part of the ICOS Cloud Continuum. The requirement for an infrastructure or infrastructure node to become part of the CC, is provided either by exploiting the ability to deploy and execute ICOS (Controller or Agent) or exploit available APIs that provide baseline functionality within the CC. In this context, ICOS provides two operation modes (based on the intent, extent, and capabilities of the infrastructure node), the ICOS Controller and the ICOS Agent mode, which depending on the scenario can be used interchangeably. ICOS through ICOS Controllers is able to monitor and maintain the topology of the created CC along with the metadata generated by the ICOS Agents that are on-boarded on the said controller. Multiple ICOS Controllers communicate using east-west interfaces to exchange local views and information.

The process for the realization of the CC can be broken down to the following scenarios:

1. ICOS Node on-boarding
   - Installation of ICOS distribution on an infrastructure node.
     - If the infrastructure node is not-restricted, then the full distribution is installed
     - If the infrastructure node is restricted, then the ICOS distribution is installed as an external virtualized component and interfaces with the infrastructure through provided APIs.
   - Assignment of ICOS Node to a controller
     - Either by manual configuration (active infrastructure deployment) or through a discovery mechanism (passive infrastructure deployment) the ICOS Node is assigned to an already running ICOS Controller.
   - Meta-information on-boarding
     - The ICOS controller retrieves from the ICOS node all the required metadata information of the following features: (i) resource availability, (ii) sensing features, (iii) data processing, (iv) HW acceleration features and (v) non ICOS nodes that are available on the ICOS node executing an ICOS agent.
   - Meta-information maintenance
     - The controller maintains the provided information in a repository along with topological information. The maintained information is frequently updated to have a near real-time awareness of all the resources availability and topology information.
2. ICOS Node removal/failure
   - Whenever an ICOS node signals its departure from the CC to the controller, the resource is removed from the repository.
   - In case of communication failure or any other deliberate failure, the node is only removed after a time threshold.

3. ICOS Controller communication
   - Each ICOS controller maintains information on capabilities, assets, resources, context concerning the subset of the ICOS nodes associated with it. This information is important for executing parts of a containerized application according to the required KPIs / Service Level objectives.
   - ICOS controllers support east-west interfaces to interconnect and expand the CC across infrastructures and domains. Two modes are supported
     - Manual configuration of peers, where known ICOS controllers are configured to talk to each other.
     - Automatic configuration where the ICOS Controllers employ discovery mechanisms in order to discover peers and interconnect with them.
   - Similarly, to the ICOS nodes failure/deletion, ICOS controllers may spawn and be removed during operations.
   - Peered ICOS controllers comprise the total of the CC by exchanging information and allowing Application deployment over infrastructure spanning a multitude of ICOS controllers and domains.

Related Roles: NP, CP, ECPP, IoTP

5.2 User Story 2 [US.2]: IoT GW/Node or device on-boarding

IoT device onboarding supports the deployment and management of IoT network gateways or devices to be instantiated as ICOS Nodes within the ICOS continuum.

IoT providers should provide at least one IoT network element to deploy the ICOS agent, so that it can be integrated in ICOS. The on-boarding of IoT infrastructure nodes in CC, in general follows the pattern described in US.1. Here we provide some specific provisions for IoT devices. The following scenarios can be observed:

1. IoT Gateway onboarding: The IoT gateway is the entry point to IoT devices from the ICOS perspective. The gateway should always run the ICOS agent, to ensure the management of resources and devices in an integrated and seamless way. The onboarding process should indicate how the gateway needs to be integrated with the other components provided by ICOS: data storage, processing nodes, etc... During the gateway onboarding process, ICOS needs to receive a response from the gateway regarding the IoT configuration options exposed to the system.
2. IoT Node onboarding: The node is an IoT device which will be collecting data and sending such data through an IoT network. In the currently rare occasion, the IoT node is not restricted and ICOS Agent can be deployed, a similar deployment process with the IoT GW will be followed. However in most of the cases an IoT node may not be able to run the ICOS agent and also not offer a public API in order to interface as a restricted device, and therefore it might not be available to the continuum (but we might consider that IoT nodes are “dumb” data providers) In this case the availability of IoT node is performed through the IoT GW which can deploy an ICOS Agent. The onboarding of a node process should indicate if there are any configuration options available to ICOS, and through which methods are usable.

Related Roles: ECPP, IoTP

5.3 User Story 3[US.3]: New Edge/Aggr/Core Resource On-boarding

As explained in User Story 1 that covers the CC realization, the infrastructure node on-boarding is also applicable here. However, this user story provides some specific details in particular for cloud (edge/aggr/core) on-boarding.

Infrastructure and resources on-boarding, including Computation, Storage and Network across Cloud, Edge (near and far edges) and IoT initially will have the following procedure:

In order to on-board an ICOS user that does not bring any infrastructure, ICOS must provide the necessary resources to meet the user's requirements. Infrastructure and resources on-boarding, including Computation, Storage and Network across the ICOS Ecosystem must follow specific criteria and be carefully selected depending on whether the related resources are assigned by ICOS for shared usage or exclusively for the specific user. ICOS will be able to discover available resources and attempt matchmaking. If the attempt fails, ICOS has the ability to spawn the resources needed on behalf of the user to ensure the alignment within the SLA. When these resources are shared among multiple users, ICOS will offer a quota in order to limit the usage of mentioned resources.

Matchmaking criteria includes features such as proximity, availability, and applicability of the resources, as well as the latency, bandwidth or network components availability.

ICOS will offer a very specific and dynamic quota to optimize resource usage and operation cost on one hand, and, on the other hand, to ensure that the quota is always respected, being rigorously checked by telemetry services.

Challenges:

- Aggregation of resources and infrastructure is possible at runtime, ICOS should be able to define a topology of resources and be able to offer available / close (GPS proximity) resources to be aggregated. For example, a map with existing edges in Barcelona, when a new edge is added to ICOS, should appear on it as well.
- Usage of shared resources brings the need of accessibility provision and control. Whether the shared resources are owned by ICOS or not they must be provisioned within authority control or be integrated with given authorization and be exploited within all means of privacy and security for all involved users.

Related Roles: NP, CP, ECPP

5.4 User Story 4 [US.4]: New Service Deployment

This user story describes the new service deployment workflow. Two scenarios are discussed, i) the baseline case - where the ICOS controller may satisfy application's resource requirements; and ii) the general case - where the ICOS controller needs to coordinate in order to satisfy the application requirements;

Scenario 1 - Resource Availability on a single ICOS controller

1. Application request arrives at the ICOS Controller
   - The application request provides to the controller the application descriptor that contains all the deployment requirements for the application components, such as required computation, communication, data sources, etc.
2. ICOS Controller, processes the Application descriptor, in order to acquire the resource requirements.
3. Based on the acquired information the ICOS Controller provides the best matching for the request.
4. The process of the business interactions required for the provisioning of the matched resources is not part of the user story

5. ICOS controller notifies the ICOS agents for the provision of resources.
   - ICOS Agents validate the resource availability and respond to the ICOS Controller
   - If resources are not available a new matching will be required.
6. ICOS controller executes the deployment of the Application components on each location, interacting with each agent. Details of each deployment are specific to the Application and technology domain at infrastructure level and are intentionally omitted.
7. By the end of the previous step, the Application is considered deployed and active, and all data sources and components are interconnected.
8. During operation, the ICOS controller keeps track of the health of the execution.
   - Fault tolerance and Migration/relocation loops are active (not detailed here) during Application operation
9. Telemetry acquired for the running application is gathered and maintained by the Controller

Scenario 2 - Resources availability via Controller Coordination

In this scenario, the application requested resources are not available within the vicinity of a single controller. As such, coordination amongst controllers is activated in order to match the requested resources. It should be noted that the ICOS controllers are distributed and dynamically organized. Each controller manages a subset of the total amount of resources (according to the organizational structure) but can interact with other controllers to combine the resources from both groups.

The main difference in this scenario is at the initial steps where the resource request needs to be referred to other controllers prior to matching decisions.

The complex service deployment user story is the following:

1. An ICOS Controller receives a request to execute an ICOS Application.
2. The controller analyses the app requirements and realizes the app cannot be successfully executed in the current context.
   - The reasons for this could be diverse. For instance, some dataset is not available in that context, or there are not enough resources to execute the application according to the agreed SLA.
3. The controller contacts other controllers (broader scope, according to the ICOS control structure) and analyses if the application can be successfully executed.
   - If not, iterate on this step until a successful solution is found.
4. After the resources are found the process of signalling the ICOS Agents is performed via each corresponding ICOS Controller
5. Rest of the scenario follows the flow of Scenario 1.

Related Roles: AppInt, AppDev, CCP, Customer

5.5 User Story 5 [US.5]: Dynamic Re-configuration / Optimisation of Application Resources

Dynamic reconfiguration of application resources should be based on appropriate AI/ML models that are trained with network telemetry data in one or more ICOS instances. Requirements and Model Construction stages:

- Collection Module: A network telemetry module that can gather information from all involved ICOS nodes of all ICOS instances. We assume that applications use resources from various ICOS instances. Data should be also gathered from application instances.
- Data cleaning and pre-processing: Collected monitoring data is pre-processed prior to the ML-training, so as to ensure proper data format and low level of noise in the training dataset.

- Model Training: Using a large part of the collected monitoring data, several local ICOS node-specific models are trained. During the training phase, several model hyperparameter (model deepness, model learning rate) configuration is considered and validated, to finally find the optimal model configuration. To support federation across local models, a periodic aggregation of multiple local parameters is performed to build a global observability model. Maximization of model performance is based on an objective function, directly related to the application optimization. Since the model supports federated learning, targets including latency reduction and/or security and privacy protection are achieved. Further, training might take place across multiple ICOS instances.
- Model Validation: To quantify the efficiency of the trained model(s), a performance metric is calculated over collected samples, not encountered during the model training phase.
- Model Deployment: The validated model is deployed in every decentralized location that comprises the ICOS nodes, optionally after model packetizing operations, such as dockerization.
- Model Inference: After model deployment, monitoring data is made available to the model(s), so as to provide corrective actions, reconfiguration activities, predictions, alarms towards achieving the application optimization objective.
- Model Operation: The output of the ML models is exploited to configure Applications and/or resources.

Related Roles: NP, CP, ECPP, IoTP, CCP

5.6 User Story 6 [US.6]: Data Access and Processing

Data access and processing refers to the ability of the ICOS system to provide access to the data generated within an ICOS instance and process it. This can be either application data of the Application running within the ICOS Instance, or data that will be later used for model training in ML-aided ICOS components for network reconfiguration and resource optimization, as described in the previous user story.

The user story assumes a state where the ICOS instance has already been fully deployed, i.e., all the ICOS Agents up and running with a proper configuration in the ICOS nodes selected to be part of the deployment. The trigger of the story is an external event either related to a data being monitored for processing (for instance, the uploading of a new piece of data, the modification of the value within a file or the collection of data from an IoT sensor) or related to the users of the Service deployed on the ICOS instance (for instance, a request to run a computation using data collected by the infrastructure). Regardless of the origin of that trigger, a request to start the processing is received by any of the ICOS Agents belonging to the ICOS Instance.

1. Upon the reception of the request, the receiver checks the permissions of the submitter to authorize the execution.
2. Decide to orchestrate its execution locally [move to step 3] or offload it onto another ICOS Agent forwarding the request [Iterates overstep 2]
3. Collection of mandatory data to start the computation [security check]
4. Split the computation into several sub-tasks
5. Orchestrate the execution of these subtasks offloading part of them onto other ICOS Agents within the same ICOS instance or invoking non-ICOS nodes. The execution of each of these sub-tasks will be treated as a completely new request.
6. Upon the completion of all the sub-tasks, the host ICOS Agent retrieves the necessary results of the sub-tasks [security check] to compute the final result.

Related Roles: NP, CP, ECPP, IoTP, CCP, ICOS End User, Customer

5.7 User Story 7 [US.7]: Deployment and Controllability from ICOS Shell

The Application Integrator should be able to fully manage the ICOS Instance throughout its lifecycle using the ICOS Shell. This should include:

- register, modify and remove infrastructures, nodes and devices;
- define, start, modify, delete application components;
- enable, disable and configure specific functionalities (e.g., discovering, monitoring, anomalies detection, deployment optimisation);
- provide runtime monitoring for the application (e.g., performance metrics, logs, security audits, anomalies, optimisation/migration suggestions/plans).

Given the role of the ICOS Shell, it enables several scenarios and it is involved in most of the user stories already presented in this section. One scenario that is worth to detail is the deployment of a new application using ICOS. It will be the most common scenario of usage of the ICOS Shell and can be described by the following steps:

1. The user downloads and installs the ICOS Shell on her laptop (or access an already installed version remotely);
2. The user creates a new "ICOS Instance" (a logical container for all data related to a given application);
3. The user specifies the resources available for the new instance;
   a. In case the user already knows some ICOS Controllers that will be used by the Instance, manually registers them (providing endpoints and credentials);
   b. If the "Discover" functionality is enabled, the user can select additional controllers discovered because they are public or because they are known to already registered controllers;
4. ICOS Shell will collect meaningful metadata from the controllers that can be shown to the user (e.g., number, capabilities and capacity of the nodes);
5. The user defines the application that will be deployed in the new instance. It includes a manifest that describes the application components and their requirements (resources, network, performance, data, security);
   a. The ICOS Shell should support the definition of the manifest providing templates and/or editors to assist and simplify the definition of this manifest using standard and well-known formats when possible (e.g., YAML, JSON). Also converters from other formats (e.g., Kubernetes, Docker Composer) could be useful;
   b. Applications can be deployed as microservices, as serverless functions or a combination of microservices and serverless functions. The format to define applications in ICOS should support all these combinations;
6. The user triggers the deployment of the application through the ICOS Shell;
   a. In more complex cases (e.g., multiple controllers registered and/or multiple components to be deployed) a matchmaking will be necessary. The ICOS Shell should support this step by providing suggestions and/or showing relevant metadata collected from the controllers;
7. The ICOS Shell will contact the selected controllers sending the application manifests and requirements. The controller will be responsible for the actual deployment;
8. After the deployment, the ICOS Shell will collect from the controllers the relevant status and monitoring data about the application that will be shown to the user (including anomalies detections, security audits).

The ICOS Shell should provide an easy and efficient access to all these functionalities without requiring advanced knowledge of the underlying infrastructural and ICOS technologies. The ICOS

Shell could have a GUI (e.g., web-based application) for most common functionalities and a Command Line Interface (e.g., kubectl[1]-like command) for more advanced and specific commands.

Beside supporting the ICOS Instance life cycle, the ICOS Shell could also provide support to: a) Infrastructure providers to on-board their resources (e.g., facilitating the installation of ICOS Controllers and Agents) and b) Application Developers during the development/integration of their applications (e.g., providing access to sdk, API documentation, testing environments, and providing DevOps-like functionalities to automate the build, distribution and deployment activities).

Related Roles: CCP, AppInt

5.8 User Story 8 [US.8]: Security and Trust Control and Monitoring

From the point of identity and access control and monitoring, ICOS system must have APIs supporting Authentication, Authorization and Audit capabilities as well as libraries supporting Authentication and Authorization to third parties.

An AppInt creates the Application Template and publishes it on ICOS. The ICOS Application Template is used to create ICOS Instance. The Application Template can be changed through time and all the changes to the Application Template as well as the changes to the ICOS Instance should be captured by the ICOS Shell and logged within the system.

Moreover, all the authorizations with the API calls pertaining to the ICOS user (API calls to the ICOS Shell) or on behalf of the ICOS user, need to be logged with the ICOS Shell. All interactions of the ICOS user with ICOS Shell should be secured and audited with the same set of libraries used throughout the ICOS Shell.

In general, all Authentication and Authorization events should also be logged.

When the Customer wishes to review audit logs, he should be able to do this through the ICOS shell. The ICOS Shell provides the ICOS user with a view of these authorizations and the usage of the API on behalf of the user.

When an AppInt needs to provide AuthT/AuthZ capabilities, he should be able to use available libraries. In this regard, the ICOS platform should provide an SDK for the AppInt to make the ICOS Instance ICOS-aware and the ICOS Instance should be able to report some application-level events to the ICOS Application Lifecycle Controller.

From the point of security control and monitoring, the ICOS system should provide anomaly detection, mitigation and recovery as well as compliance detection and enforcement.

Regarding anomaly detection, the Customer must be alerted when an anomaly on the cloud/network/edge provider is detected through the use of the ICOS notification system.

In this sense, as the data is being aggregated from the ICOS Agents the stream of data is being applied to the specific pre-trained AI model suitable for the ICOS Application Composition. Any anomalies detected by the ICOS Application Lifecycle Controller or ICOS Shell (wherever the anomaly detection process runs) should be reported using the notification system.

Next, the ICOS platform must be able to categorize anomalies and the ICOS user needs to be able to review the anomalies and enforce suggested mitigation action(s).

In this sense, anomalies detected by the ICOS Application Lifecycle Controller or ICOS Shell are being categorized (compared) and are used by the anomaly detection model. This model can be updated with the new knowledge, considering the knowledge from the ICOS users or other actors with the possibility to categorize it. This process should enable ICOS Shell to report less non-relevant anomalies. Further, reported anomalies should be linked with mitigation actions, including the enforcement of specific

[1] The Kubernetes command line tool - https://kubernetes.io/docs/reference/kubectl/

actions (e.g., patching process, change to the ICOS Application Template) over ICOS Application Composition (or ICOS Node that is part of the Application).

Regarding compliance detection, ICOS platform must be able to gather the events and map them to controls of specific standards and/or to specific policies and rules. In this sense, ICOS Agent aggregates the data coming from the ICOS nodes and the ICOS Application Lifecycle Controller maps the data to controls of specific standards and/or to specific policies and rules. ICOS Application Lifecycle Controller has knowledge of standards and controls as well as policies and rules and can be updated with the new knowledge.

Next, the Customer must be able to review events related to controls of specific standards and/or to specific policies and rules and implement necessary actions (based on policy recommendations). In this sense, the events related to controls of specific standards and policies and rules, which are reported by the ICOS Application Lifecycle Controller through the notification system, are linked with mitigation actions, including the enforcement of specific actions (e.g., patching process, change to the ICOS Application Template) over ICOS Application Composition (or ICOS Node that is part of the Application).

Related Roles: CCP, NP, CP, ECPP, IoTP, Customer

6 Use Case scenarios addressing ICOS challenges

6.1 UC1: Agriculture Operational Robotic Platform (AORP)

6.1.1 Summary

The use of agro robots for precision agriculture enables the development of farm processes, lowering operating costs, increasing awareness of technology's potential impact and digital competencies, improving resource efficiency, and lowering the amount of plant protection products used. To achieve these benefits, it is necessary to further develop digital and robotic systems based on data exchange ecosystems and services based on their semantic processing to provide knowledge and tools that will increase efficiency, ensure safety, and confirm product quality in the supply chain while reducing costs. It is also necessary to develop the cooperation of transport platforms with robots and increase the possibilities of specific activities for agricultural robots.

6.1.2 Detailed Description

Definition: The Agriculture Operational Robotic Platform (AORP) is used to implement Use Case 1. Its field tasks and missions include sowing and tending crops, removing weeds, monitoring crop development, and identifying threats. The platform moves autonomously through the field, performing the assigned missions. The robotic platform consists of control and driving modules. In addition, it is equipped with interchangeable tools - a seeder and a sprayer. The accuracy of the tasks is controlled by an RTK GPS, while the working environment is monitored using a camera on the front of the robot and ultrasonic sensors. Besides the robotic platform there is a transport platform that could act as a gateway.

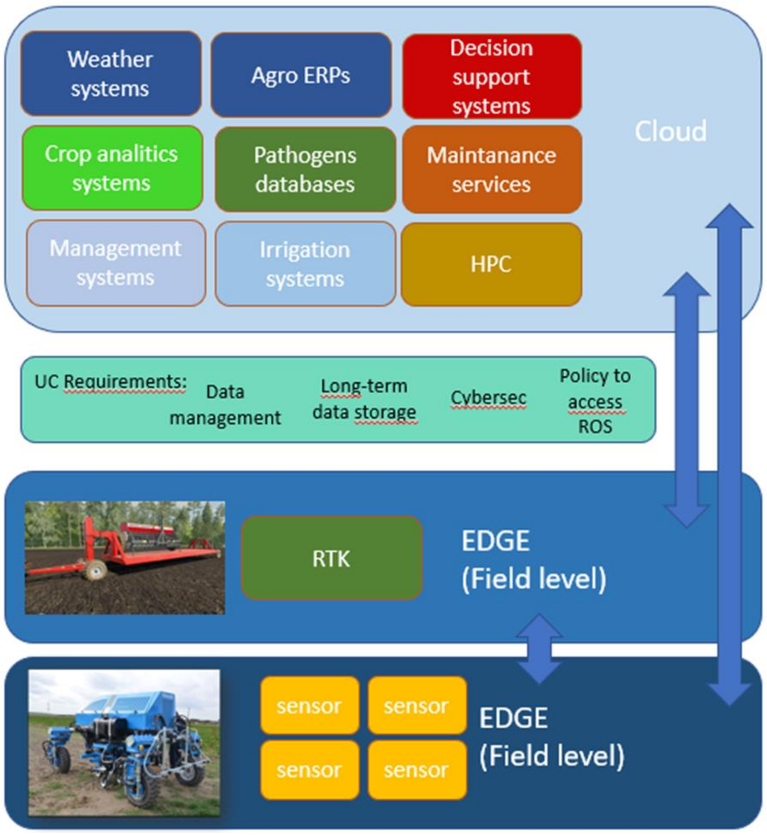


Figure 2: The Agriculture Operational Robotic Platform High Level View.

The mobile platform takes advantage of four independent steerable wheels; each wheel's rotational speed can be controlled separately. With such a mobile platform, advanced manoeuvres in the field, such as skid steering and Ackermann steering, are possible. The diesel engine on the mobile platform provides the power for hydraulic pumps and alternators to generate enough electricity to charge the batteries. These batteries are connected to a power inverter to provide electricity to external electronic equipment. The capacity of the fuel tanks is sufficient for 88 hours of continuous operation.

Autonomy: On the robot, a rugged embedded industrial computer (Neousys Nuvo) is providing computational power to processes on the robot, where an Ubuntu server is installed as the operating system, and on top of that, ROS2 is functional. A ROS2-CAN bridge node is responsible for low-level communication with the robot's ECU through a USB-CAN adaptor. The robot is equipped with RGB and RGB-D cameras, lidar, IMU, encoders, a GNSS system, etc., and its data is being processed in ROS2.

Cameras and lidar together are used for segmentation, object detection, navigation, and mapping, while taking advantage of two GNSS receivers beside global positioning provides global heading information. The data from GNSS is fused with IMU and encoders to provide more accurate state estimation.

With the raw sensor data and navigation stack on ROS2, the robot can simultaneously localize itself on the map and navigate to the destination while avoiding obstacles. The communication channels to the driving platforms (local) and the internet (cloud) would allow the robot to be controlled and/or monetised, as well as receive mission plans and transfer data to and from the cloud.

Communication with the machine will be upgraded to LoRa, WIFI, and xG modems.

Functionalities. Three main functionalities have been selected for implementation that will profit from and make use of the ICOS:

- Predictive Maintenance of the machinery and remote steering
- Crop management analytics (weed map) - main focus in ICOS
- Validation and Improvements of the ML models for robot operations and steering

Predictive maintenance of the machinery and remote steering: Data from cameras, logs, and information from all other devices installed on the robotic platform will be stored using ICOS as raw data on the cloud. Data size per day can reach 100 GB. The next step will be cloud analysis (prediction), which will take vibrations and signal control information into account. In the event of any issues, it will send back the control signals to the robotic platform. The data should be stored using data management tools in long-term storage.

There is a need to create a user interface for maintenance management, and parameters storing and control. As the robots are expensive devices, particular focus should be put on securing the connection. There should also be a way to define in ICOS the limited policy to access from the ICOS meta system.

Crop management analytics (weed map): During the first pass of the robot in the field, using the predefined mission, machinery will take field images that will be used for the purpose of creating a weed map. During the second pass the robot should already make a precision treatment based on the location on maps. The expected accuracy is 2-3 cm.

The computation will start on the edge but will need in short time any possible processing power and be able to send input images. Depending on the network it can be far edge, (to the platform), or it can be near edge . The data management tools should allow for transparent management in the continuum. Depending on the constraints (e.g. connectivity) and on the requirements (e.g. precision, e2e delay) the ICOS system will optimize the use of ICOS Elements (computing, storage, network resources) along the Cloud Continuum.

Validation and improvements of the ML models for robot operations and steering: This functionality will allow us to improve the AI models used on the near and far EDGE. The data will be sent from the devices to the cloud to train, validate and improve AI models that will be used for further missions, and for improving the robot capabilities. Usually predefined missions are being used. From a computational perspective the robot will use mostly the EDGE due to the requirements related with movement speed and the management of the unexpected obstacles or events (e.g., obstacle detection).

6.1.3 User stories

UC1 user stories, from a user perspective, are listed in Table below

Table 2: UC1 user stories

Type of user[2]	Goals
User, Robot, Transport platform	Transferring to the field: Robots are prepared, basic maintenance is done. Robot is moved from the farm to field: on wheels (fields in the immediate vicinity) (1) or on the transport platform (larger distance, using roads) (2).
User, Robot	Mission (common): Map upload, path following, task execution, monitoring, video data collection, navigation, status
Robot	Mission – monitoring/ inspection (R): Map creation, path following, task execution, video data collection, navigation, status
Robot	Mission - seeding (R): Map creation, path creation, video data collection, navigation, status
Robot	Mission - weeding (R): Map upload, path following, task execution, video data collection, navigation, status
Robot	Mission - spraying (R): Map upload, path following, task execution, video data collection, navigation, status
Robot	Re-filling (R): Established refueling/refilling point near transport platform, robot calculates low fuel/fluid level (sensor), abort the mission and calculate the path to drive to the refilling point. After refill back to the previous mission.
Robot, User	Obstacle detection (R,U): There is an obstacle in the robot's path that makes it difficult to complete the mission, mission stops, wait for user/system action
Robot	Predictive maintenance (R): Before the malfunction (detection), Robot stops
Robot	Communication lost (R): The robot lost communication at the field and connected to the driving platform. Robot and platform lost connection – store all data.
Robot	Robot 2 platform data exchange (R): After the mission is completed, the robot is loaded onto the platform, and starts sending data to the platform, after which the robot shuts down the system.
Transport platform	Data synch at farm (cloud) (P): The platform will communicate with cloud servers and exchange the data,
Common	Cloud calculations (C): NN algorithms calculations, etc.
Admin	System update (A): Process of uploading software/ module updates
Common	ML models improvements (C): AI/ML models improvements on the cloud

[2] User type: (U) – User; (R) – Robot; (P) – Transport platform; (A) – Admin; (C) – Common

6.1.4 Use Case - Scenario Requirements

- The data should be stored using data management tools in long-term storage. Data size can be up to 100 GB per day.
- There should also be a way to define in ICOS the limited policy to access ROS from the ICOS meta system.
- The data management tools should allow for transparent management in the continuum.

6.2 UC2: Railway Structural Alert Monitoring system (RSAM)

6.2.1 Summary

FGC infrastructure includes metro and commuter lines in and around the city of Barcelona, tourist mountain railways, and rural railway lines which serve more than 90 million passengers per year. Whilst most lines are conventional adhesion railways, the FGC also operates two rack railways and four funicular railways. The main lines of FGC are:

- Barcelona – Vallès, that is a suburban railway line linking Barcelona with the Vallès region, in the metropolitan area, and it consists of 53 km of track and 37 stations, serving 66 million passengers per year.
- Llobregat – Anoia, is a suburban railway line linking Barcelona with the Baix Llobregat, Bages and Anoia regions, and consists of 138 km of track and 41 stations, serving 25 million passengers per year. It also includes several freight branches.
- Lleida – La Pobla de Segur, that is a railway line linking the city of Lleida with La Pobla de Segur, in the Pyrenees, and consists of 90 km of track and 17 stations, serving 250.000 passengers per year.

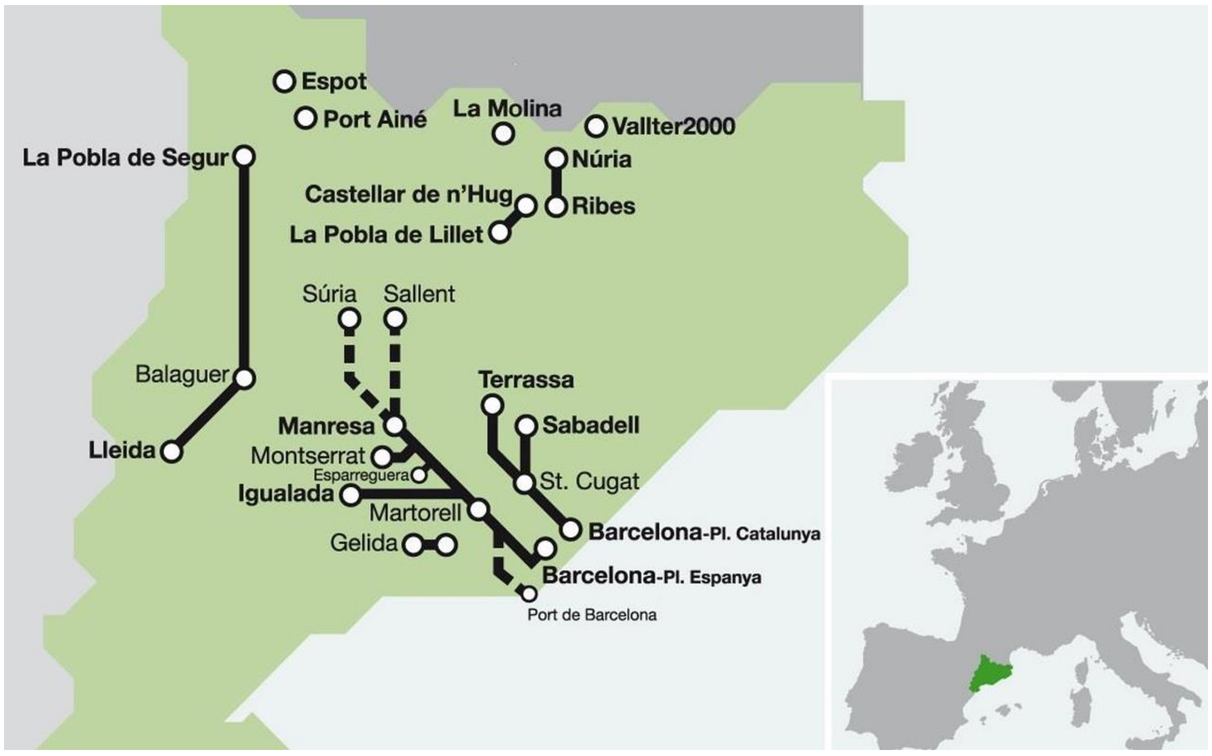


Figure 3: FGC infrastructure managed, including the three described railway lines

On all these railway lines, the massive deployment of sensors along different parts of the infrastructure is essential for the optimization and improvement of service and safety. The increasing number of sensors and its specific, and typically siloed solutions, present an increasing complexity related to the management and operations of such solutions.

Today, the railway monitoring process to improve the maintenance cycle is basic, and for most railway operators it is done preventively (once every fixed period) through a special train with sensors on its wheels which runs through the whole rail system. This special train can measure several key parameters of the railway system, such as the height difference and width between two lines, and thus identify where, potentially, corrections in the lines are needed. However, this measurement is only taken every once or twice a year; in the remaining months, nobody knows what happens (only physical inspections are available: very costly and uncommon), and moreover, there is no established procedure to evaluate the cost-effectiveness of the actions taken to address the identified rail line issues. Indeed, digital technology, such as IoT, aims to minimize the monitoring and maintenance costs by gaining knowledge of the status of key aspects of the railway infrastructure in real-time: rail tracks levelling, tensions, and slope, surrounding areas settlements and falling elements, catenaries maintenance, cyber processes monitoring, etc.

6.2.2 Description

The main challenge to be addressed by the use case is related to the continuous monitoring of critical infrastructure on rail tracks to ensure safety and improve maintenance activities.

Functionalities. Three main operational challenges should be supported by specific solutions that leverage ICOS solutions, including:

- Energy-efficient solutions for low-power IoT devices to guarantee safety operation monitoring in real time while ensuring a very long lifetime of the deployed technology in remote locations. Improve raw data transmission and balance processing between the edge and cloud.
- Improve wireless networking protocols to achieve reliable system operation in remote locations while ensuring connectivity management for the whole ICOS continuum.
- Edge to cloud orchestration of several applications according to complexity, processing, or time requirements while using the same devices deployed and improving the coexistence of real-time processing and coordination with cloud services.

The initial area to deploy and validate the use case for the Railway Structural Alert Monitoring system (RSAM) is the line in Lleida-La Pobla due to its difficult access to several of the areas of the line and its orography generating possible geological incidents.

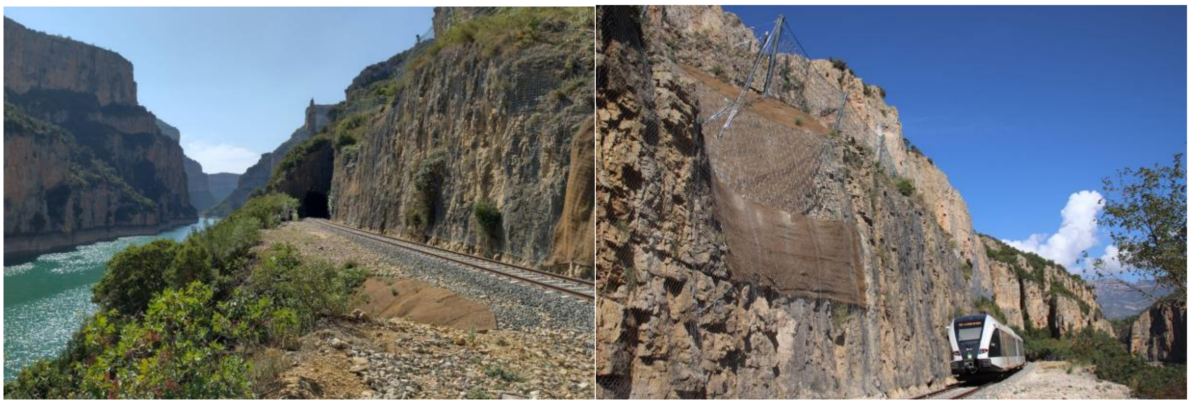


Figure 4: La Pobla railtrack and train circulation under high risk orographic environment

The line is along an area where communications may be limited in availability and bandwidth, making it possible to benefit from processing at the edge while sharing limited amounts of extremely relevant information to the upper layers of other applications. There has been an identification of different applications relevant to the use case.

- Rail track geometry: the application will support safe operations by deploying a digital and wireless monitoring system that will collect and deliver real-time information regarding the quality parameters to monitor critical infrastructure status to support the decision makers and to timely detect possible anomalies or physical threats regarding the railway track.
- Alarm detection: The alarm detection module is connected to the deployed devices, and it allows detection and acquisition of possible alarms. The detection of alarms and response actions can be required to be processed at the edge to ensure safe operations even if connectivity with upper layers is not fully available. Such response to events detected might also include the request for additional information to the physical devices to collect additional contextual data about the possible incidents and thus to better design or select a response plan.
- Maintenance: To optimize the decision-making process and to exploit all the available resources, the maintenance application will also be onboarded within the ICOS architecture. The objective of the application is to identify the trend and predict the moment when the condition where quality parameters would not be met, and therefore plan maintenance activities to mitigate such risk. The proposed application will request available resources at edge and cloud level based on connectivity and data transfer requirements and will run appropriately.

6.2.3 User stories

The RSAM (UC2), described in detail in the paragraph above, will be based on the monitoring of rail infrastructure for safety and maintenance. The UC2 will aid in the digitization of rail infrastructure operations and resource optimization, resulting in increased rail operational efficiency. UC2 user stories, from a user perspective, are listed in Table n.

Table 3: UC2 user stories

Type of user	Goals
End User (Control room operator)	I want to receive alerts of possible safety situations from the sensors deployed at the rail infrastructure as soon as they are detected. Then the operator can further explore the operational status and can take actions based on the information provided.
End User (Field geologist)	I want to display the time series of all data parameters collected from sensors at the rail infrastructure, to be integrated in the geological models and visualization tools.
End User (Field Engineer)	I want to review the alert events generated by each of the sensors, so that I can identify possible safety and quality non compliances.
End User (Field Engineer)	I want to visualize and interact with the prediction pattern for each parameter of the sensors, which have a direct effect in the maintenance activities.
End User (Maintenance responsible)	I want to receive warnings on possible quality parameters regarding non-compliance situations, so that I can plan maintenance activities. I also want to be able to visualise data from the evolution of the parameter monitored.

6.2.4 Use-case Scenario Requirements

- IoT Gateway needs to be onboarded as a processing device at the ICOS continuum
- IoT Gateway will be available as a processing node at the ICOS OS
- IoT Gateway will offer API endpoints to a limited number of configuration features of the IoT devices

- Relevant rail applications (monitoring, alarm, maintenance) will be onboarded at ICOS, and should be executed according to the requirements of data transmission and processing capacity defined for the use case
- Relevant monitoring parameters might be calculated at the edge using raw data collected by IoT devices
- Critical applications should operate regardless of connectivity availability
- Data integrity and synchronization should be ensured by ICOS

6.3 UC3: In-car Advanced Infotainment and Multimedia Management system (IAIMM)

6.3.1 Summary

In-car services are crucial for establishing a positive business case for the development of connected and autonomous vehicles. Mobility is changing, making vehicles an integral part of the customer digital world (or life): as vehicles become increasingly tech forward, the experience will expand beyond the screen to delight all the senses. The rise of autonomous and connected cars will also generate new demand for in-car infotainment and entertainment. The IAIMM Use Case will offer innovative media content and services focused on tourism to enhance the user experience while traveling in a car and getting to know and explore new places.

This service will provide the user with an in-depth tailored experience about the place they are visiting, showing the tourist extra information and knowledge around that new area in an immersive way. The possibilities of this technology will be exploited not only for interaction between users inside the car but also for others remotely.

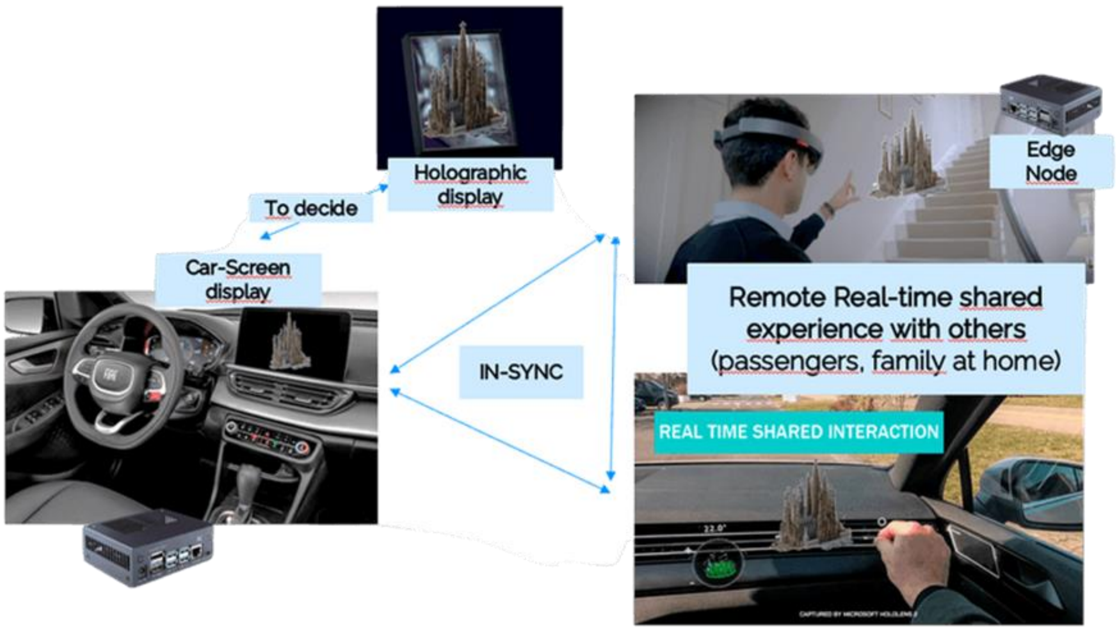

Holographic display
To decide
Car-Screen display
IN-SYNC
Edge Node
Remote Real-time shared experience with others (passengers, family at home)
REAL TIME SHARED INTERACTION

Figure 5: In-car Advanced Infotainment and Multimedia Management systems

6.3.2 Description

This ICOS solution is designed to provide to the user a mixed reality interaction service by optimizing the distribution of multimedia content (such as Videos, Interactives 3D models, Audio) and maintaining high levels of Quality of Service (QoS) and Quality of Experience (QoE). The solution also provides a secure multiuser communication and interaction infrastructure able to ensure privacy and security of shared data. This is achieved through a multilevel resource and data management system that utilizes cloud, edge, and device resources to provide location-aware services. The service aims to provide high-quality multimedia functionality for planning and enjoying trips and visiting tourist sites (Points of Interest - PoI), even in low connectivity situations. The service architecture includes nomadic edge nodes for hosting rendering and pre-processing services that ensure high quality content with low latency. Cloud nodes are used for hosting complex analytics modules, an Extended Reality (XR) manager, and a media content repository for large datasets.

The service will use real-time and dynamic information from the Car System to determine when the car will be in a specific location, considering factors such as traffic and weather. Additionally, the ICOS System will prepare in advance for when the service will be used, allowing for seamless and efficient operation. One of the key features is the Automatic Retrieve of 3D Models and Media Contents by Location (or Search). This feature allows for fast downloading of large data on-the-fly and on-demanding, making it easy to access and utilize. Additionally, the service also allows for the Visualization and Interaction of the 3D Models and Media Contents, with the capability for remote offloading at the Edge. The service also includes several collaboration tools, such as the ability for Interaction among Peers, the creation of Teams, P2P Messaging, P2P Video Calls, the ability to Share Models, and synchronized Interaction.

Lastly, the service will also be linked with car devices and sensors, allowing for events triggered by the car to impact the operational of the Service.

The following picture introduces the high-level architecture of the IAIMM service highlighting functionalities provided and software modules and where the components will be executed in a three layers deployment Cloud, Edge and End-User device.

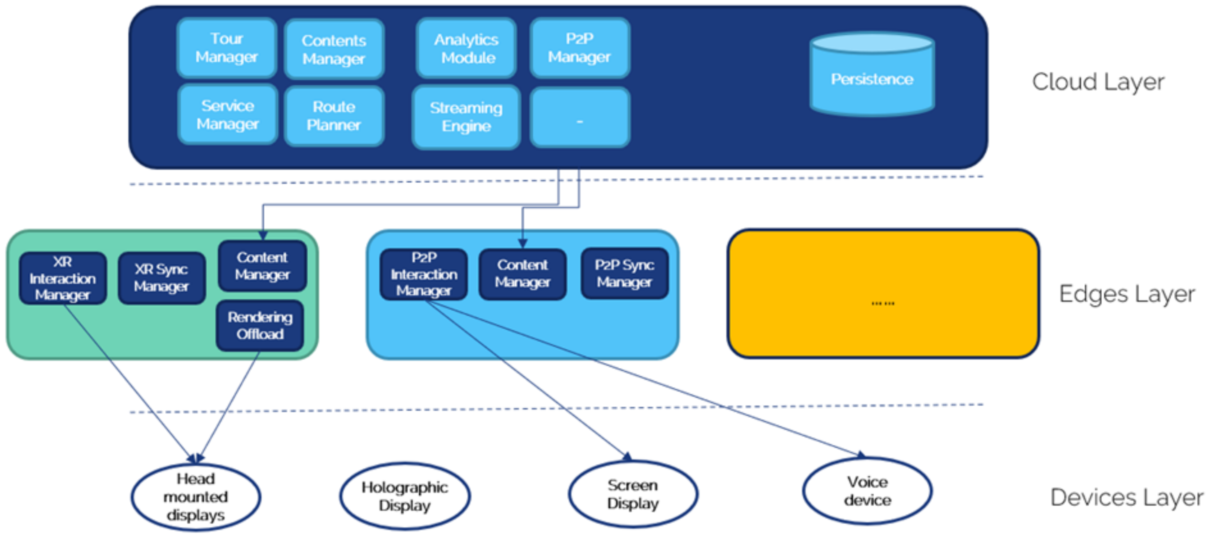


Figure 6: Three layer high-level architecture highlights the IAIMM functionalities and software modules

6.3.3 User stories

UC3 user stories, from a user perspective, are listed in the Table 4.

Table 4: UC3 user stories

Type of user	Goals
End User	I want to receive (from the car) the location of where it is, so that media content (3D Models, Videos, etc.) can pop up when the car gets close to a Point of Interest.
End User (Engineer)	I want to display (in the car) multimedia content so that the end user can visualize and/or interact with it.
End User (Engineer)	I want to interact (in the car) with the displayed content so that the user can interact with the service.
End User (Engineer)	I want to visualize and interact with high resolution multimedia content (i.e. 3D models) to maximize the QoE. The car must integrate an on-board computational node to allow the service to be nomadic (edge node) so the visualization and interaction will be with low latency and high bandwidth.
End User (Engineer)	I want to send and receive data from each client towards the server so that every client-interaction is in-sync with the others.
End User	I want to have all users be interacting with in-sync content so that the experience is being performed close to real time.

6.3.4 Use-case Scenario Requirements

- The Car system must implement an API to provide the GPS location and the speed of the car.
- The car must provide a screen where it is possible to show multimedia content (3D Models, Videos, etc). The screen may be a holographic display, a projector (to project to the window) or a flat screen.
- The car must provide a way to interact (via remote control) with the service.
- The car must integrate an on-board computational node to allow the service to be nomadic (edge node).
- The car must provide connectivity to allow other devices to access the internet.
- The on-board computational node (edge node) must be connected to the internet (4G/5G).

6.4 UC4: Energy Management and Decision Support system (EMDS)

6.4.1 Summary

The increase in economic and social activity following the emergence of the COVID-19 pandemic and the humanitarian crisis in Ukraine have generated a unique and challenging energy situation across Europe. The steep increase in the price of fossil fuels (oil and gas) has impacted the wholesale price of electricity generation, leading to a large increase in retail prices. The need to reduce fossil fuel consumption has increased demand for the production and distribution of renewable energy. The Republic of Ireland, together with many other European countries, is on a journey to reduce emissions by 51% by the end of this decade and reach net-zero emissions no later than 2050. For an island nation like Ireland, the intermittency of renewable generation, low levels of grid energy storage, and limited interconnection with other power grids are key factors that raise the importance of demand side management. Furthermore, with government incentives in place for electricity microgeneration and electric vehicle purchases, the ability to manage these prosumer and grid-level renewable assets in a way that satisfies the needs of the consumer, the grid operator, and the energy retailer is critical to reaching the 2050 zero emission goal. Artificial intelligence and smart connected devices can play an important role in allowing customers to lower their energy costs by deciding when to buy, sell, use, or store energy, while at the same time helping the grid by providing power during peak demand and mitigating curtailment.

SSE Airtricity (SSEA) is one of the largest providers of green energy in the Republic and Northern Ireland, with more than $2.5 billion invested in the development of Ireland's sustainable energy infrastructure and a large number of onshore and offshore wind farms generating more than 720 MW of renewable energy. SSEA is enabling the transition to smart homes through active participation in the SMART meter program promoted by the national network, Electricity Supply Board Networks (ESBN). Time-of-use billing and Distribution Supplier Officer/ Transmission Supplier Officer (DSO/TSO) flexibility scheme payments will further drive uptake in smart meter services especially for customers with capacity to supply power and/or shift usage (i.e., customers with microgeneration, home energy storage, EV's, heat pumps etc). In addition, SSEA supports the green energy transition with installation of solar PV, home energy retrofit and EV chargers.

The SSEA user case (UC4) aims at providing an Energy Management and Decision Support System (EMDS) using the ICOS continuum on data collected from five smart homes, see Figure 7, including the use of Machine Learning models, Edge computing and Cloud capabilities.

Figure 7: Smart Home configuration for UC4

The energy management system will generate personalized and optimized energy suggestions tailored to customer needs based on AI models and sustainable solutions. The AI will dictate when/how energy will be used/produced/stored, with hyperparameters adapting or updating through reinforcement and/or federate learning. The customers should be able to track improvements between actual costs vs optimized costs (or CO2 emissions) to build their trust in the ICOS decision-making.

6.4.2 Description

The roll out of smart meters in Ireland, initiated in 2019, is part of the national Climate Action Plan and supports the development of smart grids, microgeneration, and the transition towards low carbon emissions. Smart meters enable customer cost awareness, which can lead to the shifting of energy consumption to cheaper times of the day, providing the customer with cost savings and accurate bills based on actual usage instead of estimates. In addition, smart meters can measure the export of microgeneration and facilitate remuneration for electricity exported to the grid. Therefore, customers are or will be empowered with several possible options, including:

- Buying from or Selling to the grid (where home battery storage exists, or customers have EV's with bidirectional charging)
- Sell or trade energy with peers (P2P energy trade is still at its infancy in Ireland, with infrastructure needed for energy trading in the process of being tested/developed, this is however outside the scope of ICOS project).

- Store energy.
- Create dispatchable demand (i.e., heat water, charge the EV).

Smart meters, together with the increase of smart controllable devices, provide the necessary data granularity (data intervals of every half an hour, e.g., 48 intervals per customer per day) to obtain meaningful AI models. However, customized solutions with secure data transfers coupled with CO2 emissions linked with these transmissions, are some of the challenges intrinsic to the use of smart devices.

ICOS meta-operating system will exploit edging capabilities linked with IoT devices and cloud scalability to provide secure and more sustainable solutions by analysing energy data from five smart homes. User case participants will be selected internally by SSEA to be in compliance with Data Protection and Data Privacy regulations, using a survey. This survey will assess the presence of the optimal technologies needed to test the ICOS software and the willingness of the household to partake in a trial where automated decision making is a feature and where results will be used for dissemination and publication.

The SMART technologies to select the participant households should include the following:

- Micro-generation systems: PhotoVoltaics (PV) or Wind Turbines.
- Electric Vehicles (EV).
- Heat pumps.
- Home energy storage.
- Smart meters.

To increase the data granularity and quality, SSEA in collaboration with the ICOS project, proposes to install additional technologies such as:

- Bi-directional chargers.
- Fixed/portable energy Storage batteries.
- Occupancy sensors/inductive power monitoring clamps.

Bi-directional chargers allow electricity to flow from the Grid into the Vehicle (G2V) and from the Vehicle to the Grid (V2G), in addition surplus of energy can be redirected towards the house (Vehicle to Home, V2H). V2H enables an EV to be used like a home battery system to store excess energy. Due to the large battery capacity, a fully charged EV could support an average home for several days at a time.

Home batteries can be fixed or portable and installed as a single storage unit or coupled together to store energy either straight from the power grid or generated from renewable sources. Home batteries enable reduced energy costs by storing energy at off-peak time and discharging during the most expensive hours.

While SMART METER data are available on a day +1 basis (data collected every 30 minutes by the DSO and shared with the energy providers), inductive power monitoring clamps will allow an increased data granularity down to tens of seconds with real time communication between the clamps, the IoT devices installed into the smart homes and the ICOS operating system. Occupancy sensors can be of different natures, but in general, they detect the presence of movement within a given range and transmit the signal to a control unit. This determines if the space is occupied and switches the lights on or sends data to the thermostat to heat or cool the space accordingly. The sensor can be infrared, ultrasonic, or microwave and is used to save energy by switching on and off the connected appliances.

The data collected by these SMART technologies will be used by the ICOS continuum to derive automated decisions, which will allow the customers to directly participate in demand and supply energy distribution. The ICOS AI ‘brain’ will shape the future of the Prosumers with the aim of flattening the demand curve by removing demand on the grid at peak time and boosting energy usage at night-time. It will improve demand flexibility (e.g., the capacity of demand-side loads to change their consumption patterns on a time scale), make the electric grid more reliable (avoid grid loss), and increase the usage of renewable energy sources (mitigating wind curtailment). ICOS would, at scale, enable participation in flexibility schemes at the Distribution and Transmission System Operator level (DSO and TSO) by providing sufficient aggregated demand-side and supply capability to be of material interest to DSO's and TSO's. Finally, the cost savings from the AI will aid in promoting the use and distribution of SMART meters and time of use tariffs.

The SMART meters transfer data regularly with the network provider where communications are technically feasible. The data requirements include high levels of security and privacy. ICOS edge capability will allow a minimization of data transfers, achieving a higher level of security, low latency for real-time decision making, and lower emissions (less data traffic).

The ICOS continuum will provide valuable solutions to the user Case at data management and processing level, with Cloud/Edge infrastructure and distributed applications handling. More importantly ICOS will provide a suite of Machine Learning Algorithms and trustworthy AI Models with high level of adaptability and integration from prior learning (federate learning) able to address EU policy guidelines requirements and to ensure data security and protection.

The AI models will generate automated decisions around energy usage and storage of the five houses, for example by scheduling the charge of EV cars when this is most cost effective (i.e., at night-time) or by using green/renewable energy (from the solar panel or the wind turbines) instead of grid energy. The automated decision will generate cost savings (and consequently a reduction of CO2 emissions) that should be clearly shown to the customers in a graphical format (i.e., dashboard showing costs with/without automated decisions). In addition, the resulting energy management system will enable customers to proactively manage their energy generation and storage. In time, the use and development of ICOS Metaverse Operating System, along with smart technologies and AI models, will result in an energy paradigm shift with a more efficient, cost-effective domestic systems, where electric vehicles with V2G options, home batteries and renewable energy generating systems will enable customers to contribute in the decarbonization process and in the increase of green energy consumption.

ICOS UC4 will provide a means of validating the ICOS infrastructure and allow for the integration of green energy sources, microgeneration, minimize wind curtailment, reduce CO2 emissions and to pave the way towards 2050 net zero emissions, user stories associated with UC4 will be highlighted in the next paragraph.

6.4.3 User stories

The EMDS (UC4), described in detail in the paragraph above, will be based on energy trading/usage/storage automated decisions with the aim of improving customer habits for optimal energy consumption, cost savings and CO2 emission reduction. UC4 user stories, from a customer perspective, are listed in Table 5.

Table 5: UC4 user stories

Type of user	Goals
As an owner of PV, home battery and EV	I want to see that automated decisions are being made to get my cost of energy as close to the optimized minimum as possible.
As an owner of Photovoltaic panel	I want to use/store as much of the generated green energy as possible. The goal is to minimize energy transfer from/to the grid and optimize my energy cost.
As an owner of an EV with V2H	I want to leverage V2H in order to reduce my house electricity cost

Type of user	Goals
As a user of edge/IoT devices	I want to be prompted when I have forgotten to set schedules of my EV to ensure I have enough EV range. I want to be prompted when the EV schedule differs from an established usage pattern to ensure I have enough EV range and to maximize my car's sold/used energy.
As a user of edge/IoT devices	I want to be able to have information about my energy savings in graphical or tabular form. I want to be able to track my income from separate sources i.e., P2P, supplier, DSO, TSO.
As a user of edge/IoT devices	I want to be able to track the CO_2 intensity of the energy I pull from the grid. I want to be able to track how I have helped stabilize the grid. I want to be able to see how I have flattened my demand curve. I want to be able to track how I have helped avoid wind farm curtailment.
As a user of edge/IoT devices	I want to be able to track my cost savings if automated decisions were to be switched off. I want to be able to select between decisions I am happy to have made automatically and the ones I can decide upon (accept or reject).

In ICOS, the decision-making process will be fully automated, with no human input; however, the customers should be able to compare their energy consumption and related costs with and without the automated decisions. In the future, human input should be included into the process, with customers being able to select their level of involvement (spanning from 100% to 0% trust in automated decisions) deciding whether to accept or reject proposed decisions.

The KPIs related to the user case are shown in Figure 8. ICOS edge/cloud capabilities and a security and data management layer will aid in increasing the security and privacy of the customer data, with a prospect of 10% fewer cyber security attacks. This will be very important as UC4 concerns personal and occupancy data (as occupancy sensors might be used in the 5 selected homes, data related to the occupancy of the house will be used to build the AI models). In addition, data will be stored and processed locally, reducing the latency for real-time responses, and flattening the demand/supply curve. A 30% or more decrease in delay is expected in data transfer and processing when using the ICOS meta-operating system. By using smart data and inductive power monitoring, the data granularity and quantity will increase with time, and ICOS edge capability will increase data availability with an expected 20% more flexibility of available services with the same use of resources.

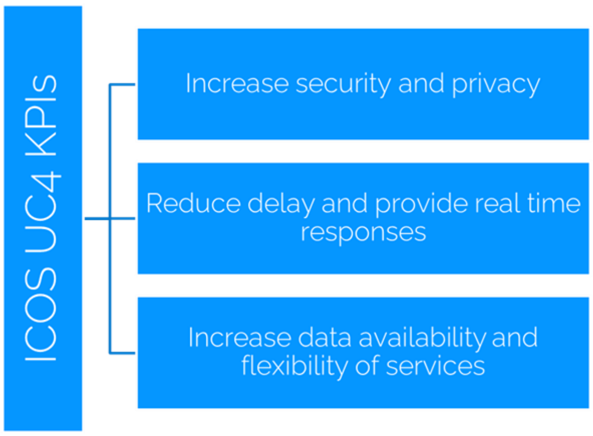

ICOS UC4 KPIs
Increase security and privacy
Reduce delay and provide real time responses
Increase data availability and flexibility of services

Figure 8: KPIs related to UC4

In summary, UC4 is an ideal test case to validate ICOS resource management and optimization, with increased speed of execution (reduced latency), greener impact (reduction of associated energy footprint) and creation/management of AI models with shorter training time, higher accuracy, and high level of adaptability/transferability to other domains.

6.4.4 Use-case Scenario Requirements

- The five smart houses should be equipped with inductive power monitoring clamps to increase data granularity down to seconds.
- The data collected from the inductive power monitoring clamps should be transmitted by fast and secure connectivity (Wi-Fi or 4/5G)
- An optimal (4 to 8 GB of RAM, Wi-Fi and fast network connectivity) IoT gateway should be available to ensure data storage, transfer, and processing at edge level.
- High-bandwidth connection should be ensured to allow fast energy decision making (sell/buy/store energy) depending on costs/time of day and to stabilize the energy curve (avoiding energy spikes due to high demand in peak times).
- High level of data security and data privacy should be ensured by the security and data management layer of the ICOS continuum as UC4 involves the use of customer and personal data with households and their occupancy easily identifiable.
- ICOS instances should be able to run/connect with cloud services.
- Suite of trustworthy AI algorithms / models conforming to policies for privacy and trustworthiness, trained in a federated learning fashion with higher accuracy, and adaptability/transferability to other domains are required to dictate the energy decision in the five smart homes.

7 ICOS Requirements Elicitation

The ICOS Requirements Elicitation is the result of analysis and a set of activities implemented within the different actors that constitute the value chain of the project. Requirements will rely on the outcomes of sections 5 and 6. The results from mentioned analysis are detailed in the following sections 7.1 and 7.2. This output is produced by following the MoSCoW method to reduce the requirements elicitation extension.

To produce the functional requirements, ICOS is analysed from five different perspectives that are considered the main topics of the project:

1. Continuum Creation: Requirements related to the onboarding and setting up of ICOS Ecosystem, including service and resource discovery and configuration.
2. Continuum Management: Requirements regarding Governance and Orchestration of the resources that compose the ICOS Ecosystem as well as the software services provided for the end users of this infrastructure.
3. Data Resiliency and Transformation: Requirements related to the ICOS internal (meta kernel) data management policies and mechanisms, including data access interfaces, caching policies, and data transformation optimizations.
4. Smart Security and Trust: Requirements related to the security and audit when using the ICOS components, detection and mitigation of anomalies as well as detection of compliance issues and their mitigation.
5. Operability Serviceability (GUI/CLI): Requirements that affect on one hand the ICOS System usability for end users in terms of graphical or command-line interfaces. And on the other hand, the operability that ICOS system offers to business application operators.

7.1 Methodology – Approach

To accomplish the requirement elicitation and provide an adequate number of requirements MoSCoW method is used. This method is based on application of prioritization strategies to include only the most qualified requirements and avoid an extremely abundant list of requirements.

MoSCoW method classifies requirements in four different premises:

- **Must Have:** mandatory requirements that add the main value to the product. These requirements cannot be missed since they compose the product. An example could be the security requirements that ensure the compliance of the product.
- **Should Have:** not mandatory requirements but also add important value to the product. These requirements have a similar impact as must have requirements, but they are not essential and can be rescheduled. An example could be performance improvement
- **Could Have:** not mandatory requirements, small impact, or none. These requirements represent non-core functionalities that are nice to have but far from essential. An example could be progressive user interface.
- **Will Not Have:** not mandatory, requirements that are left to the backlog. These requirements provide almost no impact for a specific release and will just fall out.

This method was applied separately to all requirements for each of the five topics previously mentioned. The following requirement list will only contain the first two lists of requirements, must have, and should have, the rest are discarded in order to reduce the total amount of requirements.

7.2 Functional Requirements

Table 6: Continuum Creation Requirement List

ID	Requirement Name and Description	User Story
CC_FR_01	**Resources Catalogue:** The ICOS MUST support a resource registry to record (publish) available resource to operate application workloads	US.1 Cloud Continuum Realization.
CC_FR_02	**Discovery:** ICOS MUST support methods to discover registered infrastructures	US.1 Cloud Continuum Realization.
CC_FR_03	**Topology Awareness:** ICOS should be able to monitor and maintain the topology of the created Cloud Continuum.	US.1 Cloud Continuum Realization.
CC_FR_04	**Controller Communication:** ICOS should allow the communication of multiple ICOS controllers to exchange local views, policies and information.	US.1 Cloud Continuum Realization.

Table 7: Continuum Management Requirement List

ID	Requirement Name and Description	User Story
CM_FR_01	**Smart resources first allocation and migration:** ICOS MUST be able to find a near-optimal match (considering different metrics, such as response time, energy footprint, monetary cost) in terms of nodes to run one business application taking into account nodes performance, reliability and availability	US. 1 Cloud Continuum. US. 3 New Resource On-boarding.
CM_FR_02	**Workload Offloading:** The ICOS MUST be able to distribute the workload of the application components offloading part of their computation onto other nodes of the infrastructure, and coordinate the offloaded components	US. 1 Cloud Continuum. US. 5 Dynamic reconfiguration US. 6 Data Access and Processing.
CM_FR_03	**Function execution request:** ICOS COULD provide a mechanism to request the execution of a function on the continuum being totally transparent of the device that will host the execution	US. 4 New Service Deployment. US. 7 Deployment and Controllability from ICOS Shell.
CM_FR_04	**Distributed Control:** ICOS must provide decentralized control nodes along the entire continuum, these control nodes must be distributed according to physical Edge and IoT location in order to provide closer control over specific locality or region	US. 2 IoT GW/Node or Device On-boarding. US. 3 New Resource On-Boarding.
CM_FR_05	**SLOs Definition:** ICOS must provide the capability to set an application workload Service Level Objective (SLO) by Quality-of-Service levels to be achieved. This agreement must also define the appropriate preventive to ensure SLO compliance.	US. 1 Cloud Continuum realization. US. 3 New Resource On-Boarding.

ID	Requirement Name and Description	User Story
CM_FR_06	**Monitoring**: ICOS MUST collect monitoring infrastructure-level and application-level metrics from various sources as well as provide appropriate solutions to preserve and access historical performance data. Collecting metrics from already existing monitoring tools on the infrastructures should be supported	US. 1 Cloud Continuum realization. US. 4 New Service Deployment. US. 3 New resources on boarding.
CM_FR_07	**SLO monitor to raise a corrective plan**: ICOS MUST monitor application workload SLO to raise remedial plan as well as corrective actions when QoS levels are violated.	US. 5 Dynamic Reconfiguration.
CM_FR_08	**Predictive Monitoring/SLOs violations:** ICOS SHOULD predict potential future SLOs violations analyzing monitoring metrics with ML techniques, and prepare to avoid/absorb such risk	US. 5 Dynamic Reconfiguration.
CM_FR_09	**Green Policies Monitorization:** ICOS must be able to determine when reserved resources are not used or required for proper application operation at some point in time and provide an alert system and offer an according SLO modifications	US. 4 New Service Deployment. US. 5 Dynamic reconfiguration
CM_FR_10	**Data Management:** ICOS must be able to maintain the data sources topology as well as data source types (metadata) for proper application data assignment. This includes data source selection, data source-application binding, and data access	US. 6 Data Access and Processing.
CM_FR_11	**Data Access:** ICOS must be able to provide different data access methods, including selective access as well as streaming access, according to flexible high level data access application program interfaces	US. 6 Data Access and Processing. US. 2 Device on-boarding
CM_FR_12	**Infrastructure-agnostic programming**: ICOS must provide a programming environment that abstracts service developers away from the details of the management of the devices composing the underlying infrastructure and its network topology	US. 7 Deployment and Controllability from ICOS Shell.
CM_FR_13	**Parallelism exploitation:** ICOS MUST provide a mechanism that allows service components to decompose application components into sub-components (or tasks) to enable massive/distributed parallel execution (and achieve lower response times and a better resource exploitation)	US. 5 Dynamic Reconfiguration. US. 6 Data Access and Processing.
CM_FR_14	**Model optimization techniques:** ICOS SHALL provide different methodologies for the pruning calculation of different deep neural network backbones that provides efficient sparse connectivity for modeling high-performance models on edge nodes. Distillation techniques will be used to shorten training times using the high capability of large models, the ICOS system will offer various techniques for distilling various deep neural network backbones.	US. 5 Dynamic Reconfiguration.

ID	Requirement Name and Description	User Story
CM_FR_15	**ML Scalability and maintenance:** ICOS SHOULD perform model training detached from ICOS nodes. By training the models on a separate platform with more computing power, it is possible to produce models with higher accuracy and to reduce the time required to train them. Once the models are trained, they can be deployed on ICOS nodes to make predictions and carry out specific tasks.	US. 5 Dynamic Reconfiguration.
CM_FR_16	**Federated Learning benchmarking:** ICOS MUST be able to train models over open-source federated learning frameworks. Models trained on ICOS nodes (the edge or on the cloud) will be able to share knowledge when applicable to learn collaboratively.	US. 5 Dynamic Reconfiguration.
CM_FR_17	**AI for resource management:** ICOS MUST classify infrastructure that facilitates the resource management on the network, and implements various resource management algorithms for the continuum, as well as clustering the various resources in the ecosystem with the goal of optimizing those available.	US. 5 Dynamic Reconfiguration.
CM_FR_18	**MLOps frameworks:** ICOS MUST allow the storage, retrieval and modification AI models available to be used by clients and AI-as-a-Service (AIaaS) providers from ICOS	US. 5 Dynamic Reconfiguration.
CM_FR_19	**AI Marketplace:** ICOS SHOULD use data science and MLOps frameworks such as DVC for version control and similar libraries to ensure reproducibility and proper documentation of each model built or experiment run.	US. 5 Dynamic Reconfiguration.
CM_FR_20	**Continuous learning:** The ICOS SHOULD support algorithms for continuous learning, adapting to drifts and shifts and avoiding catastrophic forgetting.	US. 5 Dynamic Reconfiguration.
CM_FR_21	**Support of multiagent models:** ICOS SHOULD support Multi Agent Models enabling resources discovery and management.	US. 1 Cloud Continuum.
CM_FR_22	**Multicluster (secure) connectivity:** ICOS SHOULD support direct (secure) networking between Pods and Services in different clusters, either on-premises or in the cloud.	US. 1 Cloud Continuum.

Table 8: Data Resilience and Transformation Requirement List

ID	Requirement Name and Description	User Story
DRT_FR_01	**Data distribution across the continuum:** ICOS SHOULD take advantage of all available devices to place data. Flexibility to adapt to different configurations (cloud-only, edge-only, cloud-edge)	US. 6 Data Access and Processing.
DRT_FR_02	**Transparent data access:** ICOS must be able to provide location and format transparent data access methods through flexible high level data access application program interfaces (API)	US. 6 Data Access and Processing.

ID	Requirement Name and Description	User Story
DRT_FR_03	**Minimization of data transfers:** ICOS MUST avoid unnecessary data movements to increase performance, reduce network congestion, and favor privacy by exploiting near-data processing	US. 6 Data Access and Processing.
DRT_FR_04	**Support for distributed/parallel execution:** ICOS SHOULD provide the integration of data management with the execution runtime to support efficient scheduling and execution of the required tasks.	US. 6 Data Access and Processing.
DRT_FR_05	**Failure recovery mechanism/management:** ICOS MUST provide the capability to restart the failing transfers of the data in case of the failures (e.g., losing the connectivity)	US. 6 Data Access and Processing.
DRT_FR_06	**Secure data exchange:** ICOS SHOULD support data preserving communication techniques between modules (interfaces) for secure data exchange.	US. 6 Data Access and Processing. US. 8 Security and Trust Control and Monitoring.
DRT_FR_07	**Data Recopilation:** Control Nodes must be able to recollect data in a scheduled or periodic way from the nodes it orchestrates	US. 6 Data Access and Processing.

Table 9: Smart Security and Trust Requirement List

ID	Requirement Name and Description	User Story
SST_FR_01	**Trust Nodes:** ICOS SHOULD be able to provide mechanisms for establishing trust procedures for the onboarding of ICOS nodes and the deployment of applications.	US. 8 Security and Trust Control and Monitoring.
SST_FR_02	**Secure comms:** ICOS SHOULD be able to exploit available encryption and isolation mechanisms to provide sufficient levels of security at the cloud continuum level	US. 8 Security and Trust Control and Monitoring.
SST_FR_03	**Tenant Isolation:** ICOS SHOULD provide mechanisms that allow for isolation of resources across multiple tenants	US. 8 Security and Trust Control and Monitoring.
SST_FR_04	**Secure & robust model training:** ICOS SHOULD provide the ability to train models in a federated learning fashion to ensure privacy and trust. Privacy can be guaranteed by training data on the edge. Robustness is assured by assessing the data against anomaly detection.	US. 8 Security and Trust Control and Monitoring.
SST_FR_05	**Providing trustable models:** ICOS SHALL use models ethically unbiased. By using these models, ICOS strives to preserve trust in the outputs of the ICOS system and provide a fair and equitable experience for all.	US. 8 Security and Trust Control and Monitoring.
SST_FR_06	**Secure infrastructure and code:** ICOS SHOULD assess security of infrastructure (e.g., running Critical Security Controls - CIS benchmarks) and system and application code (e.g. running vulnerability scanning of docker images)	US. 8 Security and Trust Control and Monitoring.

ID	Requirement Name and Description	User Story
SST_FR_07	**SecureAPI:** ICOS MUST provide API supporting AuthT/AuthZ and Audit capabilities	US. 8 Security and Trust Control and Monitoring.
SST_FR_08	**SecureLIB:** ICOS SHOULD provide libraries supporting Authentication /Authorization to 3rd parties	US. 8 Security and Trust Control and Monitoring.
SST_FR_09	**Anomaly detection:** ICOS MUST provide a mechanism for the detection of anomalies (e.g., any kind of abnormal situations, including potential security threats, that is recorded in application, system, or network logs) in the applications/services on the cloud/network/edge provider	US. 8 Security and Trust Control and Monitoring.
SST_FR_10	**Anomaly mitigation and recovery:** ICOS MUST be able categorize anomalies and recommend specific mitigation actions or recovery process	US. 8 Security and Trust Control and Monitoring.
SST_FR_11	**Compliance detection:** ICOS MUST provide a mechanism for the detection of compliance problems regarding controls of specific standards and/or specific policies and rules.	US. 8 Security and Trust Control and Monitoring.
SST_FR_12	**Compliance enforcement:** ICOS MUST provide a system to trigger infrastructure changes to ensure standard and/or policy compliance	US. 8 Security and Trust Control and Monitoring.

Table 10: Operability Requirement List

ID	Requirement Name and Description	User Story
OP_FR_01	**Resources publication and discovery:** ICOS MUST support application workloads operators to register and discover via APIs or Browser the resources will support the operational of the business application	US. 1 Cloud Continuum. US. 3 New Resource On-Boarding. US. 4 New Service Deployment.
OP_FR_02	**Topology Provision:** ICOS MUST provide the capability to graphically describe the topology and related resources needed to run application workloads including constraints, communication and security policies to apply on each node level	US. 3 New Resource On-Boarding. US. 4 New Service Deployment.
OP_FR_03	**Resources classification:** ICOS MUST be able to catalog the reliability, trustworthiness, confidence, of system resources, in order to take into consideration during the resources allocation process	US. 8 Security and Trust Control and Monitoring.
OP_FR_04	**Resource Descriptor Data-model:** ICOS MUST define a data model to be used for describing each device/node resource capability in order to be accessible by the metaOS	US. 6 Data Access and Processing.

ID	Requirement Name and Description	User Story
OP_FR_05	**Monitoring system performance:** ICOS MUST display resources and application workloads performances in real time as well as historical performance data within a graphical view. This view should allow service operator to modify the data to display	US. 7 Deployment and Controllability from ICOS Shell
OP_FR_06	**Data Processing and ML modeling:** ICOS MUST support dashboard/interface for data processing and ML modeling purposes. A concrete demonstration of the model training and inference performance will be provided via a dedicated GUI. Data processing, ML and FL platforms will be based on open-source tools to offer scalability and transparency of the deployed models.	US. 5 Dynamic Reconfiguration
OP_FR_07	**Intuitive GUI:** ICOS SHOULD provide a GUI that is structured logically and can be understood with minimum requirement of documentation by a reasonably educated (in the scope of ICOS, e.g., network administrator) human.	US. 7 Deployment and Controllability from ICOS Shell
OP_FR_08	**Interoperable CLI:** ICOS MUST provide a CLI that implements all functionalities of the GUI. This does not only allow for automated testing during development but also for integration with future, at the time of implementation unknown, components. The output of this CLI thus must be machine readable, either by default or after providing a parameter/setting an environment variable.	US. 7 Deployment and Controllability from ICOS Shell
OP_FR_09	**Granularity of the access:** ICOS SHOULD provide the possibility to define different roles in accessing to ICOS	US. 7 Deployment and Controllability from ICOS Shell

7.3 Non-Functional Requirements

The scope of this section is to define the requirements regarding the quality of the software to be delivered. The quality assurance model is based on the ISO 25010 and represented in several perspectives as follows:

- Functional Suitability: refers to the correctness of the system in terms of properties and functionalities that the system must have and comply with, also includes Functional Completeness and Appropriateness.
- Efficiency: refers to the performance of the system which compares the resources used with the amount of results or capabilities produced by the system. Usually expressed in terms of time, capacity and resource usage.
- Compatibility: refers to the capacity of the system to be integrated or interrelated with other systems or components, as well as the degree of communication and shared resources between them.
- Usability: refers to the effort needed to acquire the capacity to use the system, more specifically, learnability, operability and understandability for an individual user.
- Reliability: refers to the capacity of the system to provide a certain number of services in a specific condition during a given amount of time. Also refers to availability and fault tolerance.
- Maintainability: refers to the effort required to apply modifications to ensure software longevity. Usually expressed in terms of Analysability, modifiability, stability and reusability.

- Portability: refers to the ability of the software to be relocated to a different environment, measured by adaptability, installability and co-existence with other software or systems.
- Security: refers to the level of protection and privacy that the software can offer, including data protection and user protection. Also refers to the ability of software to ensure the authenticity of data or user.

These desired aspects of the software must be structured into Non-functional requirements and by design be covered by the ICOS Ecosystem as well as functional requirements that describe the ICOS Ecosystem itself.

Table 11: Non-Functional requirements

ID	Requirement Name	Functional requirements
NFR_1	Functional Suitability	CC_FR_01: Resources catalog CC_FR_02: Discovery CC_FR_04: Controller Communication CM_FR_04: Data Recopilation CM_FR_11: Data Access CM_FR_12: Infrastructure-agnostic programming CM_FR_14: Model optimization techniques CM_FR_16: Federated Learning benchmarking DRT_FR_01: Data distribution across the continuum OP_FR_04: Resource Descriptor Data-model
NFR_2	Performance Efficiency	CC_FR_03: Topology Awareness CM_FR_02: Workload Offloading CM_FR_03: Distributed Control CM_FR_09: Green Policies Monitorization CM_FR_10: Data Management CM_FR_15: ML Scalability and maintenance CM_FR_20: Continuous learning CM_FR_21: Support of multiagent models DRT_FR_03: Minimization of data transfers DRT_FR_04: Support for distributed/parallel execution OP_FR_09: Granularity of the access
NFR_3	Compatibility	DRT_FR_02: Transparent data access CM_FR_13: Parallelism exploitation
NFR_4	Usability	CM_FR_17: AI for resource management CM_FR_18: MLOps frameworks CM_FR_19: AI Marketplace OP_FR_01: Resources publication and discovery OP_FR_06: Data Processing and ML modeling OP_FR_07: Intuitive GUI OP_FR_08: Interoperable CLI
NFR_5	Reliability	CM_FR_05: SLOs Definition CM_FR_06: Monitoring CM_FR_07: SLO monitor to raise a corrective plan CM_FR_08: Predictive Monitoring/SLOs violations

ID	Requirement Name	Functional requirements
		DRT_FR_05: Failure recovery mechanism/management DRT_FR_06: Secure data exchange DRT_FR_07: Data Recopilation SST_FR_05: Providing trustable models SST_FR_11: Compliance detection SST_FR_12: Compliance enforcement OP_FR_03: Resource classification
NFR_6	Security	CM_FR_22: Multicluster (secure) connectivity SST_FR_01: Trust Nodes SST_FR_02: Secure comms SST_FR_03: Tenant Isolation SST_FR_04: Secure & robust model training SST_FR_06: Secure infrastructure and code SST_FR_07: SecureAPI SST_FR_08: SecureLIB SST_FR_09: Anomaly Detection SST_FR_10: Anomaly mitigation and recovery
NFR_7	Maintainability	CM_FR_01: Monitoring and Performance OP_FR_05: Monitoring system performance
NFR_8	Portability	OP_FR_02: Topology Provision

8 Cloud and Resource management survey

This section summarizes the relevant technologies, toolkits and frameworks considered for the design and development of the ICOS ecosystem. In each subsection, the basic technology is introduced while more details can be found in the sections 1, 2, 3 and 4 of the Annexes.

8.1 Continuum Management Underlying Technologies

Cloud-Edge-IoT is a dynamic architecture that requires a very specific software solution to provide service deployment, execution and facilitate the continuum. This continuum is currently very limited due to the compatibility issues with multiple components such as operating system kernel, dependencies, drivers, and other low level software layers. In the following subsection, the basic principles of the Docker Engine are described, which is a well-established technology for open-source containerization.

8.1.1 Docker Engine

Docker Engine [1] is an open-source containerization technology built on top of Linux kernel containing other components such as chroot or namespaces. It was initially based on Linux Container Engine but later migrated to specifically developed libcontainer. Docker enables containerized applications to work directly with Linux components including network and security components (network interfaces, firewall rules). Docker containers incorporate all required application dependencies, so it is sufficient and independent from lower layers of the infrastructure. Docker provides the Docker-Compose and Docker Swarm tools to define, manage and orchestrate multi-container solutions.

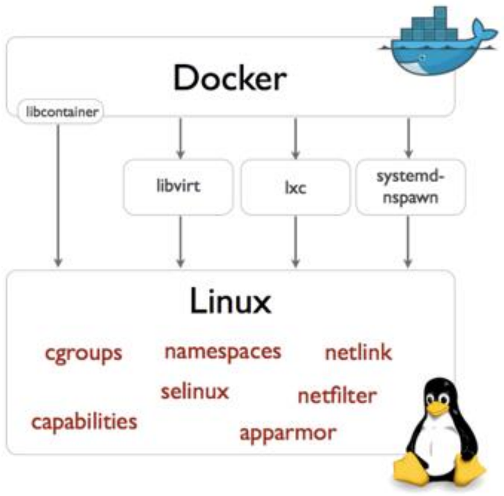


Figure 9: Docker Isolation Technology

When an application is composed of multi-container solution it is obvious that it should behave as a dynamic system since it is subject to change during its lifecycle. This includes the addition of new features/containers or replication of the existing ones. Please, mind the need to establish interconnections and communication channels between them. To accomplish this task containers must be grouped in nodes and these nodes must be managed separately. Mentioned management challenges are covered by multiple technologies. One of the most used technologies nowadays is the Kubernetes platform.

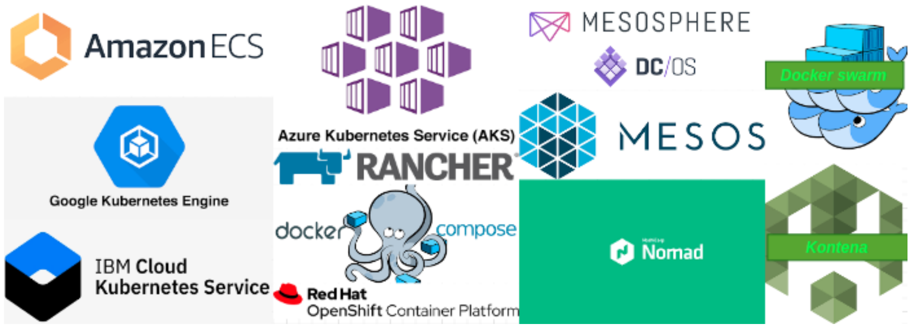


Figure 10: Container Orchestration Technologies

On the other hand, as an alternative to container-based solutions, cloud based operating systems are presented in the following section 8.1.2.

8.1.2 Cloud Based Operating Systems

A cloud-based operating system is designed for joint operation and deployment within cloud computing and virtualization environments. It is responsible for the management, operation, execution as well as all related processes of virtual machines, virtual servers, and virtual infrastructure, as well as the back-end hardware and software resources. Therefore, developers can execute all related functionalities from simple web-browsers, without the need to know in detail the underlying infrastructure. In this context, cloud based operating systems should support mobility procedures, shared resources to decouple hardware from location and maximize performance, as well as secure and efficient data storage.

In general, cloud operating systems can be classified into two major categories according to the type of access they provide to their end users: public clouds and private clouds. In the first case, storage capabilities are improved and access is unlimited. In the second case, access is limited only to certified users, such as in the case of private companies. However, hybrid models can be supported as well, where proper authentication and security mechanisms are enforced taking into advantage in parallel the improved storage capacity of private clouds.

In this context, two major deployments are OpenStack and AWS that are described in detail in the sections 1.1.1 and 1.1.2 of the Annexes.

8.1.3 Edge Containerization and Orchestration

To fully understand the continuum ICOS aims to achieve it should be considered that the edge computing for IoT follows the three-tier architecture where IoT stands at the lowest level, followed by Edge and Cloud on top of the latter. This architecture requires a technology-agnostic environment as well as strong interoperability capacities in terms of communication and interconnections between these three-tiers.

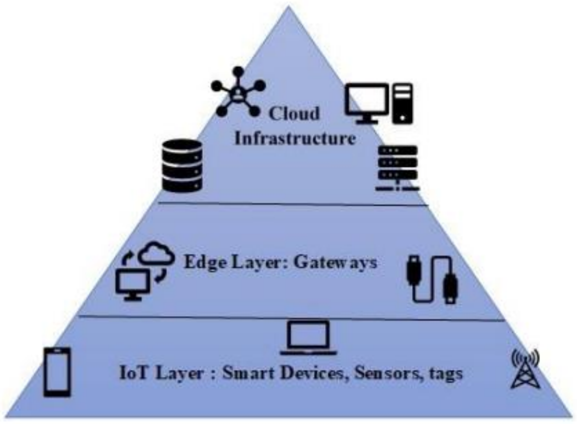


Figure 11: Container Orchestration Technologies

To achieve this environment and the capability to provide elasticity and service programmability the containerization and orchestration are used. For instance, Docker containerization technology was introduced in section 3.1. Docker alone provides a containerization for a specific application or a set of applications, but it will not provide sufficient orchestration capacities for highly distributed or multi-tenant environments such as edge computing for IoT. Consider that Edge and IoT nodes are grouped in sets of devices linked to a gateway in a particular location and together they form a cluster. [107]

Cluster Manager usually refers to a software that runs a stack of cluster nodes that it manages. This software usually provides a cluster management agent or multiple agents. These agents run on each node of a cluster to provide administration, configuration, and operation for a set of services that the mentioned cluster provides. Normally, this kind of software provides a graphical user interface or command-line interface to be able to manage clusters and to apply the mentioned set of operations and lifecycle related processes for services provided by the cluster.

To bring some light on referred set of operations included into cluster management the following cluster lifecycle operations are displayed:

- Creating a new cluster
- Removing a cluster
- Updating the control node and compute nodes of a cluster
- Maintenance and updates to the node of a cluster
- Upgrading the cluster

Several commercial and open-source cluster management technologies are reviewed in the section 1.2 of the Annexes, such as Kubernetes or K8s, Lightweight Kubernetes or K3s, OKD, OpenShift, Rancher, and Knative, OpenCluster Management. Not all of them are presented in this document since they are very numerous.

8.2 Networking and Secure Communications

8.2.1 Submariner [2]

An important aspect in IoT networks is the ability to interconnect a vast number of heterogeneous topologies, in an efficient and secure way. To this end, a commonly known tool is Submariner, which delivers direct networking connectivity between Pods and Services across different Kubernetes clusters (either on-premises or in the cloud), by providing:

- Cross-cluster L3 connectivity using encrypted or unencrypted connections
- Service Discovery across clusters

- Support for interconnecting clusters with overlapping CIDRs

The service discovery model is built following the proposed Kubernetes Multi Cluster Services [3] by the Kubernetes Multicluster Special Interest Group.

In the same context, device interconnection can be established via mesh networking. A mesh network comprises a type of local area network (LAN) topology, where multiple devices or nodes are connected in a non-hierarchical manner, so that they can cooperate, and provide significant network coverage to a wider area compared to the area coverage achieved by a single router. As mesh networks consist of multiple nodes, responsible for signal sending and information relaying, every node of the network needs to be connected to another via a dedicated link. Since mesh networks leverage a multi-hop wireless backbone formed by stationary routers, they can connect both mobile and stationary users. Mesh networks have significant advantages such as fast and easy network extension, self-configuration and self-healing and reliability as a single node failure does not result in total network failure.

More details on Submariner and mesh networking can be found in the Annex.

Reference: Kubernetes Multi Cluster Services

8.3 Far Edge Device Orchestration and Management

A significant issue in modern IoT networks is data collection and management when data sources are not in the close proximity of data centers. Depending on the devices' capabilities, far edge processing can be supported, where in this case mobile nodes can collect and process data. Hence, overall system's latency can be significantly reduced, while at the same time workload in centralized servers is reduced. However, apart from the data processing part, another important aspect is device and data heterogeneity. Hence, the main goal of far edge device and orchestration tools is on one hand to provide lightweight capabilities of data aggregation coming from diverse sources, and on the other hand to provide an easy way for device manipulation. The most important tools towards this direction are identified, such as:

- Genie [4] is an open-source distributed job orchestration engine developed by Netflix. Genie provides REST-ful APIs to run a variety of big data jobs like Hadoop, Pig, Hive, Spark, Presto, Sqoop and more. It also provides APIs for managing the metadata of many distributed processing clusters and the commands and applications which run on them. Genie[5] provides scalable, federated job and resource management for users of computational resources. From the perspective of the end-user, Genie abstracts away the physical details of various (potentially transient) computational resources (like YARN, Presto, Mesos clusters etc.). It then provides APIs to submit and monitor jobs on these clusters without users having to install any clients themselves or know details of the clusters and commands. Administrators will use the configuration APIs to register clusters and the commands/applications that run on them with Genie. The Genie nodes can have all the clients pre-installed on them or Genie will download and install them at runtime if properly configured. Users can then look up what clusters and commands are available and submit jobs to be processed. Once jobs are submitted, users can query Genie for job status and output. A big advantage of this model is the scalability that it provides for client resources. This solves a very common problem where a single machine is configured as an entry point to submit jobs to large clusters and the machine gets overloaded. Genie allows the use of a group of machines which can increase and decrease in number to handle the increasing load, providing a very scalable solution.
- Nuvla [6] is an edge and a container management platform built upon open-source software and open standards. The Nuvla platform allows you to configure any number of Container-as-a-Service (CaaS) (e.g. Docker Swarm, Kubernetes) endpoints. This means you can mix and match public clouds, private clouds and infrastructure, as well as edge devices running NuvlaEdge software. This enables a simple and effective edge-to-multi-cloud solution. The platform is application centric, hardware agnostic, cloud neutral and container native. This allows end users to manage any containerised application across a fleet of edge devices and container-orchestration engines.

- HPE GreenLake [7] is an edge-to-cloud platform offering for on-premises and hybrid workloads. The solution is fully managed and consumed via a metered, pay-per-use model at the edge, in the data center and colocation facility.
- openEuler (formerly EulerOS) [108] openEuler is an open-source operating system. The current openEuler kernel is based on Linux and supports Kunpeng and other processors. It fully unleashes the potential of computing chips. As an efficient, stable, and secure open-source OS built by global open-source contributors, openEuler applies to database, big data, cloud computing, and artificial intelligence (AI) scenarios. In addition, openEuler community is an open-source community for global OSs. Through community cooperation, openEuler builds an innovative platform, builds a unified and open OS that supports multiple processor architectures, and promotes the prosperity of the software and hardware application ecosystem.
- AWS IoT Greengrass [8] is an open-source Internet of Things (IoT) edge runtime and cloud service that helps you build, deploy and manage IoT applications on your devices. One can use AWS IoT Greengrass to build software that enables your devices to act locally on the data that they generate, run predictions based on machine learning models, and filter and aggregate device data. AWS IoT Greengrass enables user devices to collect and analyse data closer to where that data is generated, react autonomously to local events, and communicate securely with other devices on the local network. Greengrass devices can also communicate securely with AWS IoT Core and export IoT data to the AWS Cloud. Users can use AWS IoT Greengrass to build edge applications using pre-built software modules, called components, that can connect edge devices to AWS services or third-party services. One can also use AWS IoT Greengrass to package and run software using Lambda functions, Docker containers, native operating system processes, or custom runtimes of your choice.

As in the previous subsections, more details per tool can be found in the section 3 of the Annexes.

8.4 Function Management – FaaS

When deploying in full scale a cloud based IoT system, it is important to decouple application and service functionality from hardware equipment. With such an approach the physical hardware, virtual machine operating system, and web server software management are all handled automatically by the cloud service provider. This allows developers to focus solely on individual functions in their application code. Therefore, the concept of function as a service (FaaS) can be supported. In this context, two important tools as Apache OpenWhisk and Open FaaS can be considered:

- Apache OpenWhisk [9] is an open source, distributed serverless platform that executes functions in response to events at any scale. OpenWhisk manages the infrastructure, servers and scaling using Docker containers.
- OpenFaaS [109] is an open source serverless function engine that enables developers to deploy event-driven functions and microservices to Kubernetes without repetitive, boiler-plate coding. It can be deployed to multiple environments such as mentioned Kubernetes or docker. Currently it is compatible with several programming languages and contains multiple built in functions for developers.

A more detailed analysis of the aforementioned tools is provided in section 4.2 of the Annexes.

8.4.1 AI-assisted continuum management

In this section, we outline the primary technological benchmarks for continuum management, beginning with an overview of the federated learning (FL) baselines that have evolved through time and discussing the significance of federated learning in ICOS. There are several technologies for FL described, including libraries and frameworks that may be employed. We also offer a number of model optimization techniques that are connected to the ICOS intelligence layer, such as pruning, network distillation, hyperparameter tuning, and data management clustering techniques that have been appended to Appendix for AI-assisted continuum management. Then we include transfer learning, as it will be included in the AI analytics section of the intelligence layer. These techniques would cover the potential to offer ML training and inference as a service, as well as other capabilities like network optimization, data augmentation, and tuning. Finally, a list of possible technologies for the AI model marketplace is described as similar use cases.

8.4.1.1 Federated Learning Benchmarking: Baselines on the continuum and open-source tools

Federated learning is a machine learning technique that allows models to be trained on decentralized data sources, such as a large number of devices in a network. This approach enables the use of large amounts of data for training while preserving privacy by keeping the data on the devices. It is a useful approach for training machine learning models on decentralized data sources, such as a large number of devices in a network. This approach has several advantages over traditional centralized training methods, including:

- Privacy: Federated learning allows models to be trained on decentralized data sources without compromising the privacy of the data.
- Scalability: With federated learning, the data can be distributed across many devices, which can enable the use of much larger datasets.
- Efficiency: Federated learning allows training models in a distributed fashion, allowing for the reduction of the amount of data transfer and communication required, especially in cases where the data is distributed across devices with limited connectivity or bandwidth.
- Robustness: FL allows the model to be trained on a diverse range of data sources, which can improve its robustness and generalizability.

A more detailed description of FL has been presented in the section 5.1 of the Annexes, including federated learning benchmarks, baselines and open-source FL frameworks that could operate in the continuum.

8.4.1.2 Transfer Learning

Transfer-Learning has the capacity to reduce the number of computational resources required to train a model while maintaining its accuracy, including on mobile and edge devices, making it a powerful tool for enhancing the ICOS AI intelligence layer. This happens because the models have already been trained on datasets that are larger and different from the ones that will be used, allowing them to concentrate on the fine-tuning stage with the local data, which, because it is much smaller, may allow them to inherit the features that have already been trained on the task of the relevant edge device. This might provide faster training than starting from scratch as compared to conventional training approaches. We have added some state-of-the-art methods and approaches in the section 5.2 of the Annexes, describing potential tools to be used.

8.4.1.3 Model Optimization Techniques

These techniques are used to improve the performance and efficiency of machine learning models. These techniques can be applied to trained models to reduce their size, improve their accuracy, or make them more efficient for running on different devices. The section 5.3 of the Annexes contains a summary of the different methodologies for model optimization and compression techniques like pruning, quantization, and knowledge distillation that will help reduce the size and parameters of a model and make it efficient to run on edge devices.

8.4.1.4 MLOps

In this section, we describe the MLOps pillars and goals, as well as the main structure to follow within an efficient AI ecosystem. Secondly, we also describe different machine learning operation tools (MLOps tools) that are commonly used in the open-source community. These methods include all possible practices related to the automation of processes, such as the deployment, monitoring, and management of machine learning models that allow teams to work collaboratively. Finally, we describe a set of real-life use cases for common use cases where ICOS could potentially be used.

8.4.1.4.1 MLOps pillars and goals[11]

To develop models at a large scale and have a successful MLOps platform, there are four different pillars and roles for MLOps , as shown in the Figure 12, that every MLOps system should consider:

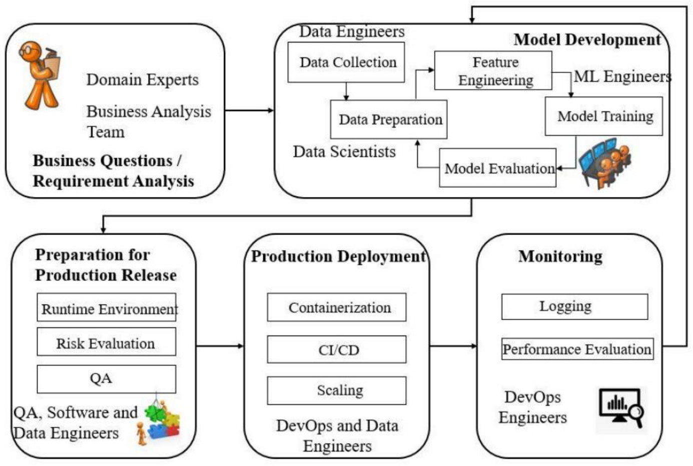


Figure 12: MLOps pillars and associated roles

- **Model deployment & experimentation**: Model deployment is an essential machine learning procedure that links models, allows appropriate API access, and streamlines operations between model and production environments. To make sure the model is operating as intended and continues to function properly in production, it goes through a number of stages including data extraction, validation, preparation, training, assessment, and testing. The performance of the model can be enhanced for usage in production by guaranteeing the validity of the data and having a faster deployment procedure.
- **Model monitoring**: Understanding the reasons for the behaviour of a model requires monitoring its performance over time and in various contexts. However, subpar performance can expose businesses to serious risks and push them to make bad choices. To understand erroneous or unreliable forecasts, a monitoring feature should enable detailed evaluation of data drift over time. This enhances the precision and dependability of the model's predictions while reducing the risks.
- **Production deployment**: This model deployment pillar focuses on automating a number of processes, such as model updating, troubleshooting, and triage for low confidence levels, model approval by interested parties for subsequent cycle phases, and model updates and manufacturing that allow models to be changed without upsetting the entire cycle. To increase the efficiency and effectiveness of the models, containerization and scaling are also involved.
- **Preparation for Production release**: This pillar relates to the version control that also entangles automated documentation for related changes, as well as a complete tracking line for all updates and changes on the models' cycle. It also includes the risk evaluation for each cycle.

Because the goal of the MLOps tools is to make the ML life cycle easier and faster, the diagram below depicts the main tools that can be used to complete one or more of the MLOps pillars.

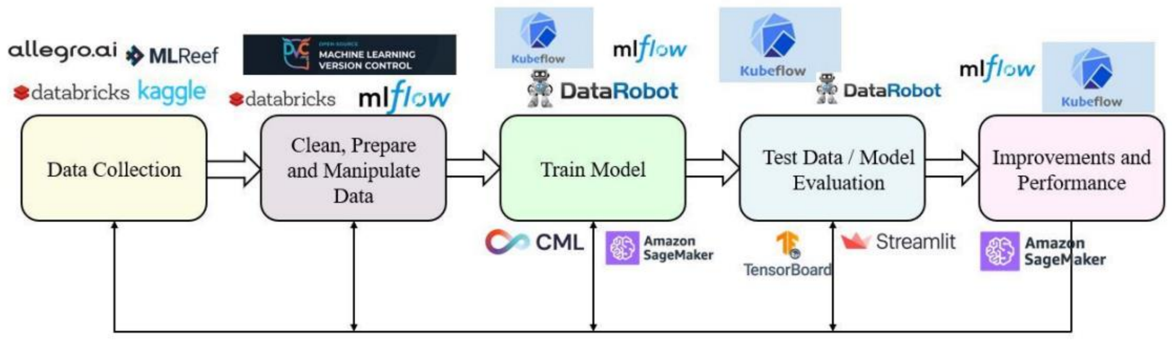


Figure 13: MLOps main operations for any of the four pillars

8.4.1.4.2 MLOps Tools

ICOS should be able to offer a solution that may assist organizations and users of the OS in effectively delivering high-quality models and enhancing teamwork between development and operations. When examining the various baselines to be used in the continuum with the goal of evaluating the technologies that are most appropriate for Machine Learning Operations (MLOps) in ICOS, the accessibility, integration with current systems, usability, security and privacy, and cost are taken into account. Building, deploying, and managing machine learning models in production should be made easier and more automated with the aid of an efficient MLOps tool. A list of the potential technologies that could be applied in this regard can be found in the section 9 of the Annexes.

8.4.1.5 AI Models Marketplace

It is an online marketplace where AI models are available to be used by clients and AI-as-a-Service (AIaaS) providers from ICOS. AI models are used in AIaaS applications and can be offered in the form of an "AI models marketplace". The AIaaS marketplace is an online marketplace where AIaaS providers can offer their models and services to AI developers and customers. Its goal is to manage, store and maintain pre-trained ML Models, the Codebase and data (locally and using the Marketplace), while providing an easy-to-use interface for users to upload and retrieve machine learning models, allowing users to create a marketplace for their machine learning models, and allowing users to build a community to share machine learning models.

- Exploration of open-source version control software that could potentially be used for the AI marketplace:
  - **HuggingFace Hub**[12] [Open-source]: It is a platform where more than 65k models have been shared as well as 6k datasets and 6k demos of projects. This hub contains GitHub-inspired features for code-sharing and collaboration. The hub offers data versioning as well, commit history, diffs, branches and around 12 library integrations.
  - **DVC** [13] [Open-source]: DVC (Data Version Control) is an open-source version control system specifically designed for data science and machine learning projects. It is similar to other version control systems, such as Git, but it is optimized for handling large files and complex file dependencies, such as datasets, models, and code.
  - **ML Metadata** [14] [Open-source]: A standard and consistent method of storing and accessing data is provided by the open-source library ML Metadata (MLMD), which captures and retrieves metadata for machine learning (ML) operations. It enables users to store metadata about algorithms, models, parameters, settings, and performance measurements, as well as input and output data. To enhance ML model performance and spot potential problems, this metadata can be retrieved and examined.

- **Mercurial** [15] [Open-source]: Any scale of software development project can be managed with Mercurial, an open-source version control system. It enables several developers to work concurrently on the same codebase without erasing their changes or creating disputes. Developers can work individually with Mercurial, a distributed version control methodology, and then merge their changes with the main codebase whenever they are prepared. This promotes effective and flexible collaboration.

- **Marketplace examples**
  - **Modelzoo** [16]: The OpenAI Model Zoo is a collection of pre-trained AI models and algorithms that are available for use and exploration by the community. It includes models in a variety of domains and tasks, such as natural language processing, computer vision, and reinforcement learning.
  - **SingulartityNet** [17]: Distributed platform created to sell AI services using tokens (AGIX)
  - **AWS Marketplace** [18]: A curated digital catalogue that you can use to find, buy, deploy, and manage third-party software.
  - **Genesisai** [19]: ML protocol containing an AI marketplace.

9 Continuum Data Processing, Distribution and Storage

Data processing and management is a complex task that, depending on the scenario and application domains, might require different approaches and communication models to be in-place in different levels of the continuum. As an example, Information Technology (IT) systems have focused on the capability of storing and querying data over the network, usually achieved via databases (e.g., SQL, NoSQL, key-value, etc.) that provide all the necessary management for data at rest. Conversely, Operation Technology (OT) systems have focused on the capability of distributing data over the network, usually achieved via publish/subscribe message bus that provide the necessary management for data in motion.

Recently, a convergence between IT and OT has started to be witnessed in several application scenarios, since they provide complementary benefits. On one hand, IT systems (e.g., Cloud) proved to be economically convenient. On the other hand, OT systems (e.g., factory processes) proved to be more scalable and reactive over distributed and constrained network resources. In addition, IT systems are increasingly embracing reactive and event-driven architectures typically seen in OT systems, and OT systems are embracing processing of huge volumes of data (e.g., data analytics) typically targeted by IT systems.

This convergence between IT and OT is hence creating an increasing need for a unified way to handle data-at-rest (storage, queries) with data-in-motion (publish/subscribe) across the entire continuum (i.e., from the Cloud to the Edge, and even Fog devices), while supporting decentralization, scalability, efficiency, and location transparent access to geographically distributed data.

In the remainder of this section, we will describe existing technology for Data Processing,Data Distribution, and Data Storage.

9.1 Data Processing

In the continuum, data processing is needed in three different scenarios. In the first scenario, called sense-process-actuate, one of the devices of the infrastructure senses an event and the IT platform is expected to process it and provide an appropriate response even activating some of the actuators to produce some physical effect. Whereas in the first scenario, data processing is triggered by an eventual data generation, the second scenario considers the processing of a continuous generation of data (stream). This continuous processing aims at keeping the data stored in the system updated or producing other streams of data with the results. In contrast with the other two scenarios where the processing is triggered by data, the third scenario considers data-at-rest; computations are triggered by the system administrator or the end user to run compute-heavy processes to analyse large amounts of data for instance to train ML models or running large simulations. In general, the currently available software development solutions target one of these scenarios and focus on tackling the specific challenges set out by the scenario they deal with; however, this technological heterogeneity incurs a wide range of programming frameworks and models that hinders the efficient development of solutions targeting complete solutions of the continuum.

The latter scenario, where data is at rest, is the traditional case of batch jobs in classical distributed systems. There are plenty of solutions in the bibliography building on the concept of task-based workflows; while some of these models are domain-specific and defined to target a specific problem (e.g., deal with collections of data [20] or run data analytics [21]), other models are general purpose [22] [23]. Some of these tools are described in further detail in the section 6.1 of the Annexes.

A stream of data could be processed using a workflow manager; however, that would raise a lot of tasks and generate a lot of management overhead. Dataflows Managers[3,4] [24] [25] – section 6.2 of the Annexes has more details on some of them – are a more efficient approach to process streams.

Finally, services falling into the sense-process-actuate scenario usually turn to Function-as-a-Service solutions to support their operation. Section 8.4 already discusses the most popular frameworks delivering Function-as-a-Service; besides those, some frameworks [26] [27] – details in Section 6.3 of the Annexes – natively support some of the programming models discussed above to handle their efficient execution across the Continuum.

Table 12: Data Processing Technology Comparative.

Software	License	Sense-process-actuate	Streaming	Batch
Colony + COMPSs	Apache 2.0	yes	yes	yes
FuncX + Parsl	Apache 2.0	yes	no	yes
Zenoh-flow	Apache 2.0	no	yes	no
Apache Beam	Apache 2.0	no	yes	yes
Apache Airflow	Apache 2.0	no	no	yes
Dask	BSD 3-Clause	no	no	yes
Spark	Apache 2.0	no	no	yes

Nowadays, Machine Learning (ML) is growingly used in broad fields of Information Technology (IT), providing novel ways of processing, and analysing data in quick and effective fashion [28]. In this context, ML, as part of Artificial Intelligence (AI) techniques, is exploited also in Edge Computing to relieve the processing burden in the Cloud infrastructures, considering the continuously increasing insertion of IoT devices in various everyday applications, either personal and residential or business [29]. As a result, such intelligent analysis mechanisms are promisingly utilized in the current Continuum Computing endeavours, attempting to achieve the optimization of load balance between Edge and Cloud. Several ML techniques have been introduced in the IT scenery and can be involved also in Edge Computing.

Table 13: Machine Learning Technology Comparative.

Software	License	Feature
Scikit-Learn	BSD 3-Clause	Appropriate for traditional machine learning algorithms due to its efficiency and simplicity.
TensorFlow	Apache 2.0	High flexibility for training and testing, as well as easy scalability for building and deployment on deep learning.
TensorFlowLite	Apache 2.0	Minimalistic, lightweight, fast and efficient version of TensorFlow for mobile and embedded devices.
River	BSD 3-Clause	Targets online machine learning and it can be used for ad hoc tasks like concept drift detection and computing online metrics.
PyTorch	Modified BSD	Appropriate for deep learning and includes dynamic computational graph and tensor-based computation.
MXNET	Apache 2.0	Suitable for research prototyping and production. Supports both symbolic and imperative programming. It is fast, scalable, and flexible.

[3] https://airflow.apache.org/

[4] https://beam.apache.org/

Software	License	Feature
Keras	Apache 2.0	Runs on top of TensorFlow. Highly used in research for building and training deep learning models.
dislib	Apache 2.0	Inspired by scikit-learn. Builds on COMPSs to run on any kind of heterogeneous distributed infrastructures.

To improve ML performance metrics, data can undergo several pre-processing operations than can be also performed with the previously mentioned frameworks[5]. We have also provided a more detailed description on the section 7 of the Annexes.

Finally, the section 8 of the Annexes, Data Pre-processing Operations, contains an in-depth analysis of such techniques. The most common operations applied to structured data[6] are:

- Data Cleansing: establishing a set of data quality rules for standardization and the eradication of duplicates.[30], [31], [32].
- Data Splitting: splitting the data correspondingly on training, validation and testing, including stratification techniques based on the requirements or the framework.
- Feature Tuning: Scaling and normalizing numerical values, impute missing values, clip outliers, and make adjustments to values with skewed distributions.
- Feature Transformation: represent data in both numeric and categorical forms and establishing conversion mechanisms like one-hot encoding, categorical counts and low-dimensional vectors [32].
- Feature Selection: determining the most important features of the dataset while removing redundant features without incurring much loss of information [33][34].
- Feature Construction: Creating new features using mechanisms such as polynomial expansion and feature crossing[35].

When working with unstructured data, in the case of images[7], text [36] or audio [37], deep learning can be used for feature extraction.

9.2 Data Distribution

9.2.1 Apache Kafka [38]

Apache Kafka[8] is an open-source message broker that aims to provide real time, low latency manipulation over data-streams. It is mainly used to manage data-in-motion and setting up data pipelines usually used in micro-services architectures as an exchange system. Apache Kafka has various architectural components for distributed streaming, see Figure 14: Apache Kafka architecture.

- Topic: In Kafka data is stored in the form of topics. Producers write their data to topics, and consumers read the data from these topics.
- Brokers: A Kafka cluster comprises one or more servers that are known as brokers. A Kafka broker can contain multiple topics with different partitions. A unique ID is used to identify each broker in the cluster.
- Consumers and consumer groups: A consumer reads data from the cluster. When a consumer is ready, it pulls the data from the broker. A consumer group refers to a group of consumers that pull data from the same topic or same set of topics.

[5] https://spark.apache.org/docs/latest/ml-features

[6] https://cloud.google.com/architecture/data-preprocessing-for-ml-with-tf-transform-pt1

[7] https://pytorch.org/vision/main/transforms.html

[8] https://kafka.apache.org/

- Producers: A producer publishes messages to one or more topics. It sends data to the cluster. Whenever a Kafka producer publishes a message, the broker receives the message and appends it to a particular partition. Producers can choose to publish a message to a partition of their choice.

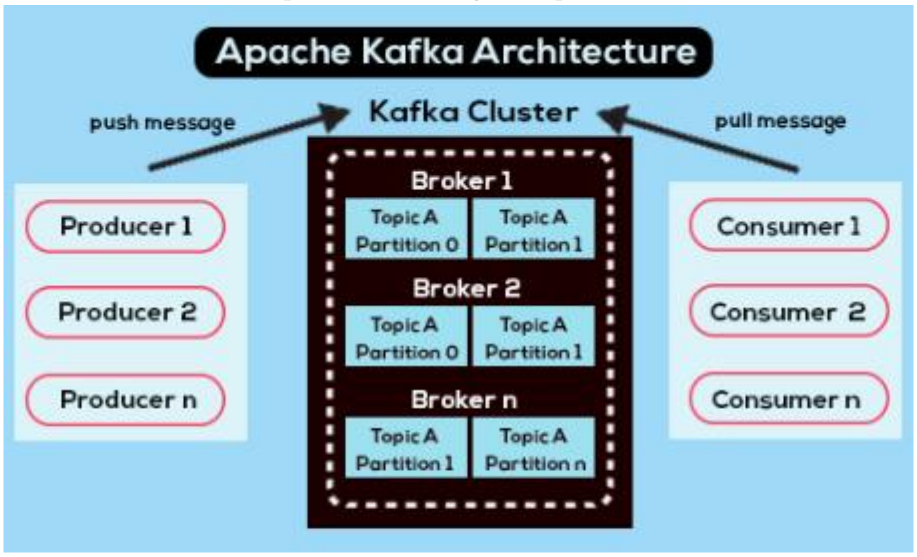


Figure 14: Apache Kafka architecture

While Apache Kafka is a solid distributed messaging platform, it has some limitations. It does not address the data-at-rest scenario or distributed computations. It is seen that Kafka lacks a full set of management and monitoring tools, and usually it is integrated with other technologies to provide full enterprise support which augments the complexity of its deployment.

9.2.2 MQTT

MQTT (Message Queueing Telemetry Transport)[9] is an OASIS standard that was initially popularised as the protocol for the IoT. It is a Client Server publish/subscribe messaging protocol based on TCP/IP protocol. There are many MQTT agents available, they vary in their functionality and some of them implement additional functionality. The main open-source agents are: Active MQ, Eclipse Paho, Eclipse Mosquitto, and Hive MQ, just to mention some. While MQTT and its different implementations are currently used in diverse, health, energy, IoT applications, etc., it brings some restrictions and limitations. MQTT architecture is broker-based protocol. The end devices (i.e., clients) communicate via a broker, they can act as publishers of a given topic, and publish events. The MQTT broker can be installed in any machine in the cloud. Similarly, applications will subscribe for an event with the broker. Due to this subscription, when there is a change in any event or parameter, the broker will notify to the subscribed clients the change in event or parameter.

[9] MQTT 3.1.1 specification, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

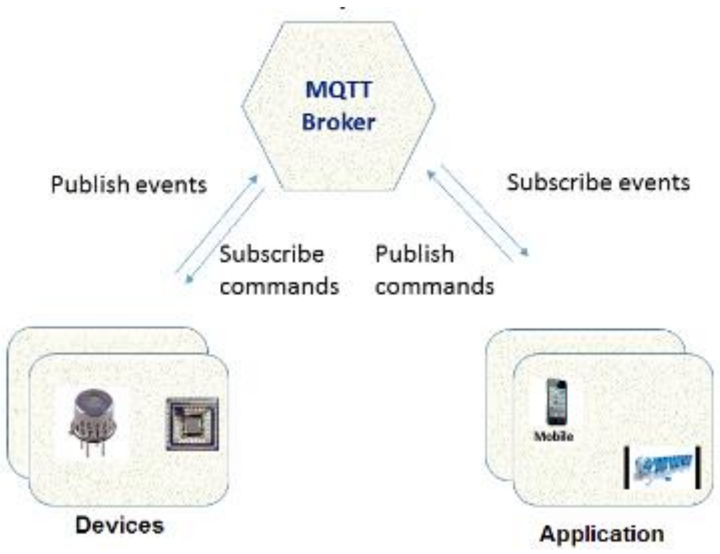


Figure 15: MQTT architecture

Its main limitation lies in that every message is send to and distributed by the message broker (server), moreover peer-to-peer communication might be a problem, and to achieve p2p it is necessarily to use a dedicated topic for every connection between two clients, but the messages would still have to pass through a message broker. In addition:

- MQTT requires TCP/IP, thus making it inconvenient for constrained and low powered networks
- The choice of topic-name has an impact on bandwidth usage
- Push mode only. This does not work with battery constraint devices that have duty cycles and missed streams of data while sleeping.

MQTT lacks support for request/reply pattern natively, if you need request/reply you need to do it into a layer above the protocol itself defining right topics and payload to create a matching/correlation logic between request and response.

9.2.3 Eclipse Zenoh [39]

Eclipse Zenoh is an extremely efficient and fault-tolerant data distribution protocol, inspired on Named Data Networking (NDN) paradigm, capable of operating across the entire continuum (from powerful Cloud servers and down to extremely constrained devices and networks). Zenoh unifies the access to any kind of data, both data-in-motion, data-at-rest, and computations. To do so, it blends traditional publish/subscribe mechanisms with geographically distributed storage, queries, and computations, while retaining a level of time and space efficiency that is well beyond any of the mainstream stacks.

The key differentiators of Zenoh are:

1. Cloud to the Microcontroller Communication. Zenoh can work efficiently and perform from server-grade hardware and networks to the embedded microcontroller and extremely constrained networks. Therefore, Zenoh allows data to freely flow vertically and horizontally from the microcontroller to the datacenter. Likewise, it provides developers a single solution for data distribution, in spite of the need to integrate technologies to bridge the communication between the enterprise and the embedded world.
2. Data Centricity and Location Transparency for Data in Movement and at Rest. Location transparency is a consequence of data centricity – in these systems users only need to express interests without any concern on the location of its source. This feature is extremely important as it makes it easier to deal with scale, failures, and load-balancing. Beyond publish/subscribe mechanisms that already leverage on location transparency for data-in-motion, Zenoh brings location transparency for data-at-rest, allowing queries to be expressed without any concerns on the

actual location of data (i.e., the location of the databases or storages). Zenoh takes the responsibility of identifying the optimal set of databases/storages available in the network where the query should be executed.

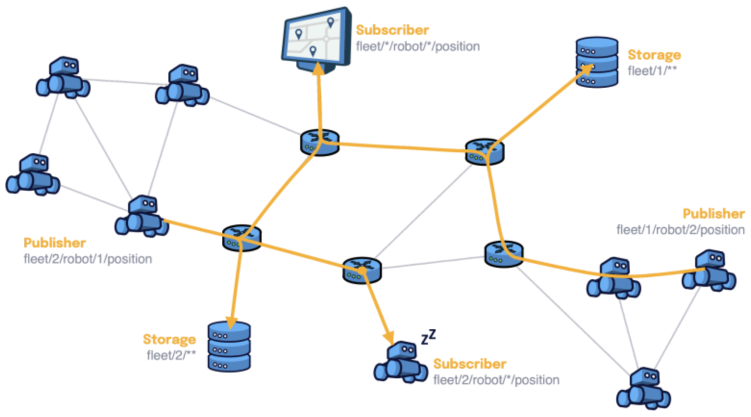


Figure 16: Zenoh topology example

3. Easy to Use and Performant. Zenoh has been designed ground up to be simple to use and high-performance across all its applicability range. It constantly delivers higher throughput and lower latency and achieves that through a simplistic and easy to use API. This ensures that developers can be productive from day one, without need to write brittle and unmaintainable code to get performance.
4. Energy Efficient. Nowadays, computer networks are consuming an incredibly amount of energy. A considerable part related to the data distribution from devices to centralised data centres in the Cloud. Zenoh was designed for energy efficiency by not only exploiting decentralized data distribution and its support for exploiting locality in communication, but also introducing minimal wire-overhead of 4-6 bytes.
5. Anywhere in the networking stack. Zenoh is agnostic to the underlying technology, implementing a networking layer capable of running above a Data Link, Network, or Transport layers of the OSI stack.

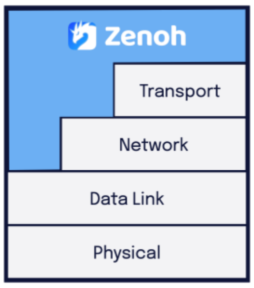


Figure 17: Zenoh layer over OSI stack

6. It runs over any topology. Zenoh can operate under different models, allowing it to run over any topology and anywhere across the continuum. It can run in (i) a peer-to-peer fashion way, allowing the creation of clique or mesh topologies; (ii) a brokered fashion way, where nodes have only a limited set of functionalities and rely on the network to provide the full Zenoh capabilities; (iii) routed fashion way, where nodes act as software routers that forward Zenoh messages between nodes.

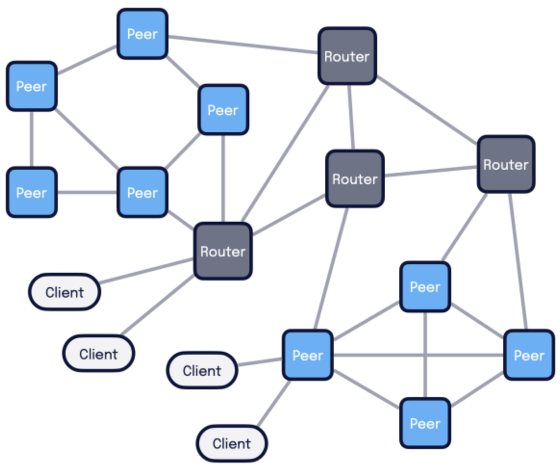


Figure 18: Zenoh entities and interconnection possibilities

Summing up, Zenoh can be integrated with today's systems to enable and support large scale distributed systems: integrate OT protocols with the datacenter world and/or integrate IT protocols with the embedded world. Thus, it offers the capability to seamlessly bridge IT and OT protocols across systems and networks.

The key challenge for protocols for the IoT is the opportunity for single point of failure due to client-server star topology message flows. The challenge has always been generally known but deprioritized to move quickly and capitalize on the real value of IoT which is centralized data storage to have interesting and valuable outcomes. Ideally, we want more intelligent nodes that have node-to-node awareness. Perhaps the most viable protocol that will help us mature to this ideal model is Zenoh which is an Eclipse Incubation project[10]. Zenoh's unified vision avoids the patchwork design in what concerns data distribution capabilities by unifying all data management procedures. Systems will natively be built as large-scale distributed systems and natively integrate with embedded and microcontroller devices.

9.3 Data management

Different groups of solutions have appeared in the last few years to deal with the data management needs in edge-to-cloud environments.

On the one hand, we find distributed databases, which focus on providing performance and scalability, and have been adapted to edge-to-cloud scenarios. Usually, they are able to efficiently manage time series data but they still require a stable distributed environment in order to operate. Examples of this kind of products are MongoDB[11] or Cassandra[12], which can also store data in other kinds of data structures. InfluxDB[13], or Druid[14] are distributed data stores specifically designed for time series data. All these databases allow fast ingestion and query of large collections of event data, but require a distributed infrastructure that is homogeneous and relatively stable, such as a cluster in a datacenter. As a consequence, they are not suitable for running at the edge, thus requiring continuous data transfers to and from the cloud, which increase latency and network congestion.

[10] https://www.embedded.com/four-key-application-protocols-for-the-internet-of-things-iot/

[11] https://www.mongodb.com/use-cases/internet-of-things

[12] https://academy.datastax.com/use-cases/internet-of-things-time-series

[13] https://www.influxdata.com/

[14] https://druid.apache.org/

On the other hand, other data management solutions are able to operate on constrained devices at the edge and are prepared to deal with the heterogeneity of these environments: devices with diverse capabilities, as well as different kinds of data. Several commercial products, such as McObject eXtremeDB[15], ObjectBox[16], or Azure SQL Edge[17], as well as open-source products such as Apache IoTDB[18] provide support in this setting, thus reducing the amount of data transfers from edge to cloud and vice versa. However, although they implement fault tolerance mechanisms, they assume a fixed infrastructure.

An important additional characteristic of the edge-to-cloud continuum is that the available infrastructure may unexpectedly change during operation: devices may unexpectedly join and start contributing with their capabilities or their data, or they may leave, either because they are replaced, because they move, or because they run out of battery. To the best of our knowledge, dataClay[19] is the only open-source solution that is able to cope with this dynamicity.

The Table 14 summarizes the main characteristics of data management solutions in edge-to-cloud. A brief summary of the open-source ones is provided in the next subsections.

Table 14: Main characteristics of data management solutions in edge-to-cloud

Software	Licenses	Data Type	Edge	Cloud	Dynamicity
InfluxDB	MIT	Time series	No	Yes	No
Apache Druid	Apache 2.0	Time series	No	Yes	No
eXtremeDB	Commercial	Arbitrary	Yes	Yes	No
ObjectBox	Commercial	Arbitrary	Yes	Yes	No
Azure SQL Edge	Commercial	Arbitrary	Yes	Yes	No
Apache IoTDB	Apache 2.0	Time series	Yes	Yes	No
dataClay	BSD-3	Arbitrary	Yes	Yes	Yes

9.3.1 InfluxDB

InfluxDB OSS is a local time series database developed by InfluxData. Its aim is to simplify the management of time series data for application developers in the context of IoT. Its main features are fast ingestion of high volumes of time series data (e.g., through MQTT or Kafka), as well as efficient query by means of a SQL-like language. It also provides useful functionalities such as dashboards for data visualization, or integration with Prometheus to ingest and manage monitoring metrics.

Although the features of its open-source product are limited in the context of edge-to-cloud environments, it can be used in combination with its commercial InfluxDB Cloud by means of its Edge Data Replication Feature. This functionality allows local InfluxDB instances to collect and process data for local use and replicate it to the cloud to aggregate and store data for long-term analysis.

[15] https://www.mcobject.com/extremedbfamily/

[16] https://objectbox.io/

[17] https://learn.microsoft.com/en-us/azure/azure-sql-edge/

[18] https://iotdb.apache.org/

[19] https://www.bsc.es/dataClay

9.3.2 Apache Druid

Druid is a column-oriented distributed data store. It is designed to quickly ingest big quantities of event data and provide low-latency queries on top of the data, and it is commonly used in business intelligence-OLAP applications.

Druid can be deployed in big clusters, processing each query in parallel across the different nodes. Regarding data ingestion, it can ingest data from files in batches, or it can be connected to a stream by means of Kafka. In both cases, ingested data becomes immediately available for querying.

9.3.3 Apache IoTDB

Apache IoTDB is a database with high performance for data management and analysis, deployable on the edge and the cloud. Similar to InfluxDB and Druid, it is designed to rapidly ingest time series data, and optimized for its processing.

Apache IoTDB can work in combination with distributed data processing engines such as Apache Hadoop, Spark, and Flink.

Apache IoTDB can be deployed in different configurations: it can either run locally on small devices such as Raspberry Pi, or the small devices can hold a client and connect to a server located in a bigger machine. In both cases, IoTDB can transfer the data files to another IoTDB instance in the cloud for backup. An alternative deployment is having the client in a small device and connect to the server in the cloud. However, this setting does not enable processing any data at the edge.

9.3.4 dataClay

dataClay is a distributed data store that enables applications to manage objects in the same format they have in memory, avoiding transformations. This enables it to deal with any kind of data structure that can be defined in an application.

dataClay was initially designed for HPC infrastructures, with the aim to optimize data-intensive applications. To this end, dataClay backends can execute arbitrary code so that data is processed within the datastore, transferring only the result instead of the entire dataset to the application (which may be running in the same node or in another one).

dataClay has been adapted to the constraints of edge-to-cloud environments by making it lightweight to run on small devices such as Raspberry Pi or Jetson boards, being able to combine them with a bigger infrastructure such as a cluster, or a server. Additionally, it can ingest/export data from/to MQTT or Kafka.

dataClay is also integrated with COMPSs for distributed and parallel processing.

9.3.5 9.3.5 Elasticsearch

Elasticsearch is an open-source search engine based on Apache Lucene for all types of data. This includes textual, numerical, geospatial, structured, and unstructured. It is distributed, fast, and scalable. Data is stored in a form of schema-free JSON documents. Official clients are available in Python, Java, .NET, PHP and many other languages. Elastic stack comes with other useful tools like Kibana for management and data exploration, Beats as lightweight data forwarding agents, and Logstash for data processing and transformation on the fly. This stack of tools is also often used as a central server for logs. Grafana is often used for building dashboards and interactive visualizations on top of the data stored and indexed in Elasticsearch. Elasticsearch will be used, among others, as a part of the ICOS Security Layer as a log management solution.

9.3.6 Grafana

Grafana is an open-source analytics and interactive visualization web application. It integrates connectors to many data sources, like Elasticsearch, MySQL, InfluxDB, AWS CloudWatch, PostgreSQL, Prometheus, and many more. Grafana is used to transform and visualize data, build dashboards, and trigger alerts. Visualizations are interactive, providing users with better insight into the data. Alerts can be sent to email addresses, Slack, Microsoft Teams, and other channels. It also supports advanced features like access control and user management. Grafana can be extended with its plug-in system. Free plugins are available in a public plug-in library. Plugins extend the list of supported data sources, visualizations, and other features. Grafana will be used, among others, as a part of the ICOS Security Layer as a log anomaly detection management, visualization tool, and alerting tool.

10 ICOS related State of the Art on research projects

10.1 Literature (related existing white papers/ open-source research)

We will refer to specific sub-areas of state-of-the-art in cloud and edge computing, which we see as possible additions to ICOS. From the research point of view, not so many papers are pertaining to the continuum, hence this subdivision is useful. In section 10.2, we refer to more such complete frameworks, which are not always state of the art research, but exist as combined CC management packages.

The cloud and edge computing sub-areas considered in this report are orchestration, task and resource allocation, execution offloading, serverless and event-driven computing, and security and data privacy.

10.1.1 Orchestration

Pegasus: Pegasus can be used to automate edge-to-cloud science workflows and the workflow provenance data collection capabilities of the Pegasus monitoring daemon enable computer scientists to conduct edge-to-cloud research. [40]

George: George, is an end-to-end general purpose Long-Running Applications (LRA) scheduler by leveraging the state-of-the-art Reinforcement Learning (RL) techniques to intelligently schedule LRA containers. The Authors present an optimal container placement formulation for the first time with the objective of maximizing container placement performance subject to a set of operation constraints. [41]

Kraken: Kraken is a workflow-aware resource management framework that minimizes the number of containers provisioned for an application DAG while ensuring SLO-compliance. The authors design and implement Kraken on OpenFaaS and evaluate it on a multi-node Kubernetes-managed cluster. The extensive experimental evaluation using DeathStarbench workload suite and real-world traces demonstrates that Kraken spawns up to 76% fewer containers, thereby improving container utilization and saving cluster-wide energy by up to 4x and 48%, respectively, when compared to state-of-the art schedulers employed in serverless platforms. [42]

Scanflow-K8s: A practical platform, so-called Scanflow-K8s, that enables autonomic ML workflows on Kubernetes clusters based on the Scanflow agents. MNIST image classification and MLPerf ImageNet classification benchmarks are used as case studies to show the capabilities of Scanflow-K8s under different scenarios. The experimental results demonstrate the feasibility and effectiveness of the proposed agent approach and the Scanflow-K8s platform for the autonomic management of ML workflows in Kubernetes clusters at multiple layers. [43]

10.1.2 Task and resource allocation

HERTI: HERTI is a reinforcement learning-augmented system for efficient real-time inference on heterogeneous embedded systems. HERTI efficiently explores the state space and robustly finds an efficient state that significantly improves the efficiency of the target inference workload while satisfying its deadline constraint through reinforcement learning. [44]

Zenith: Zenith is a novel model for allocating computing resources in an edge computing platform that allows service providers to establish resource sharing contracts with edge infrastructure providers apriori. Based on the established contracts, service providers employ a latency-aware scheduling and resource provisioning algorithm that enables tasks to complete and meet their latency requirements.[45]

Energy-Aware Resource Scheduling for Serverless Edge Computing: The authors present energy-aware scheduling for Serverless edge computing. Energy awareness is critical since edge nodes, in many Internet of Things (IoT) domains, are meant to be powered by renewable energy sources that are variable, making low-powered and/or overloaded (bottleneck) nodes unavailable and not operating their services.[46]

Polaris Scheduler: Polaris Scheduler is a scheduling framework with respect to edge sensitivity and service level objectives. The scheduler gathers information about the workload, to optimally schedule the cluster and infrastructure at the edge. To do so, it focuses on capturing the inter-dependencies of workloads, enabling continuous monitoring and dealing with properties of infrastructure nodes. The Polaris scheduler can be implemented on any Kubernetes cluster and can then be further customized with extension plugins to enhance the scheduling process.[47]

EMCS: By deciding based on completion time, cost and energy consumption, the Energy-Efficient Makespan Cost-Aware Scheduling (EMCS) algorithm will schedule tasks at cloud and edge. The EMCS algorithm is based on an evolutionary strategy, it learns how to schedule which tasks, from biological reproduction by implementing concepts like mutation and crossover. The authors show the advantages of EMCS in the cloud and in the edge over other methods used for the same purpose. [48]

StaSA: Service–Time-Aware Scheduling Algorithm (StaSA) is built on dynamic pod scheduling, enabling enhanced ant colony optimization and optimizing the task scheduling process at the edge, maximizing the node utilization by increasing the CPU and memory utilization and lowering the pod service time. In experiments the authors show that STaSA is outperforming similar methods for these criteria. [49]

Pilot-Edge: Pilot-Edge is a distributed resource management abstraction and framework along the Edge-to-Cloud continuum. It decouples the resource management and the workload management and, through an easy to use FaaS API, Pilot-Edge allows tasks to be moved to different parts of the Edge-to-Cloud Continuum. The authors addressed different challenges like heterogeneity and dynamism, and investigated different topics like data collection and model training for which Pilot-Edge is well suited.[50]

10.1.3 Offloading

Resource Allocation for NOMA MEC Offloading: a third strategy, hybrid NOMA, which contains the strategies of OMA and pure NOMA. As the main contribution, the authors analytically characterize the optimal resource allocation, i. e., the joint power and time allocation, for two-scheduled-user cases. Simulation results show that the proposed resource allocation method in hybrid NOMA systems yields lower energy consumption and delay than the conventional OMA scheme. [51]

A Multi-Agent Deep Reinforcement Learning Approach for Computation Offloading in 5G Mobile Edge Computing: A multi-agent deep reinforcement learning (MADRL) based decentralized cooperative offloading decision algorithm, which determines whether the computing tasks are executed locally or placed on an appropriate edge node to minimize system costs. [52]

pSFC: pSFC provides a fine-grained SFC deployment scheme in the PDP (Programmable Data Plane) to tackle the problem. The authors first model network functions as control flow graphs (CFG) and the process of deployment as a one big switch (OBS) problem, and then propose an ILP (Integer Linear Programming) model for resource optimization for the OBS problem, which is NP-hard. To solve this problem efficiently, pSFC first composes multiple SFCs for eliminating redundant resources, decomposes the compound CFG based on the resource limitation per stage, and finally maps OBS into the substrate network. [53]

10.1.4 Serverless and Event-driven computing

Serverless4IoT: Serverless4IoT is designed to simplify the process of designing, deploying and maintaining serverless applications over the IoT Cloud-Edge Continuum. It integrates a SERVERLESS4IOT Orchestrator, responsible for the allocation of serverless functions and resources and a SERVERLESS4IOT Gateway as a local broker to enable serverless computing. In contrast to the FaaS platforms offered by big cloud, SERVERLESS4IOT supports the deployment of ad-hoc software stack and does not promote vendor lock-ins. [54]

Apollo: Apollo is a modular and distributed runtime system for serverless function compositions on cloud, edge, and IoT resources. Building on a flexible model for applications and resources, as well as a base of many independent agents, Apollo is able to distribute processing and the implementation process itself. Thereby, Apollo reaches a high degree of parallelization and is able to fine-grain the distribution of the applications. [55]

10.1.5 Security and data-privacy

Citadel: Citadel, a scalable collaborative ML system that protects both data and model privacy in untrusted infrastructures equipped with Intel SGX. Citadel performs distributed training across multiple training enclaves running on behalf of data owners and an aggregator enclave on behalf of the model owner. Citadel establishes a strong information barrier between these enclaves by zero-sum masking and hierarchical aggregation to prevent data/model leakage during collaborative training. [56]

Egeon: This paper presents Egeon, a novel software-defined data protection framework for object storage. Egeon enables users to declaratively set privacy policies on how their data can be shared. In the privacy policies, the users can build complex data protection services through the composition of data transformations, which are invoked inline by Egeon upon a read request. As a result, data owners can trivially display multiple views from the same data piece and modify these views by only updating the policies. And all without restructuring the internals of the underlying object storage system. [57]

ELSA: Edge Lightweight Searchable Attribute-based encryption system (ELSA). ELSA leverages the cloud-edge architecture to improve search time beyond the state-of-the-art. The main contributions of this paper are as follows. First, the authors present an untrusted cloud/trusted edge architecture, which optimizes the efficiency of data processing and decision making in the IIoT context. Second, they enhance search performance over current state-of-the-art (LSABE-MA) by an order of magnitude. [58]

Lasagna: Lasagna is an SGX oriented DNN inference performance acceleration framework without compromising the task security. Lasagna consists of a local task scheduler and a global task balancer to optimize the system performance by exploring the layered structure of DNN models. The experiment results show that the author's layer-aware Lasagna effectively speeds up the well-known DNN inference in SGX by 1.31x-1.97x. [59]

OTE: Optimal Trustworthy EdgeAI (OTE) allows developers to implement more trustworthy sensing, perceiving and different modeling techniques. The authors presented OTE as a solution for smart cities and argue that for example traffic monitoring, human flow monitoring, and environmental challenges can take advantage of the new OTE research pipeline. The pipeline provides increased IoT intelligence, trustworthy AI and the increased cloud intelligence which is leading to advances in the whole span of IoT-Edge-Cloud continuum. [60]

10.2 Research Projects

10.2.1 mF2C

Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem (mF2C) is an EU H2020 research and innovation program funded research project (2017-2019) which develops a global management framework for the fog-to-cloud ecosystem. It has been designed as an open, secure, decentralized, multi-stakeholder management framework, and includes novel programming models, data storage techniques, service creation, brokerage solutions, SLA policies, resource orchestration methods, as well as guaranteeing privacy and security.

The mF2C framework has been designed in a multi-layered hierarchical architecture, where nodes on the top are the most powerful (the cloud) and nodes on the bottom are more constrained devices, including IoT devices. Fog nodes are organized by areas (at layer 0) and controlled by a leader (one of the most powerful nodes in the area, at layer 1). A group of nodes at layer 1 (leaders from layer 0) are grouped and controlled by a leader (one of the most powerful nodes in the area, at layer 2). And so forth up to the cloud layer.

The mF2C control architecture is the following:

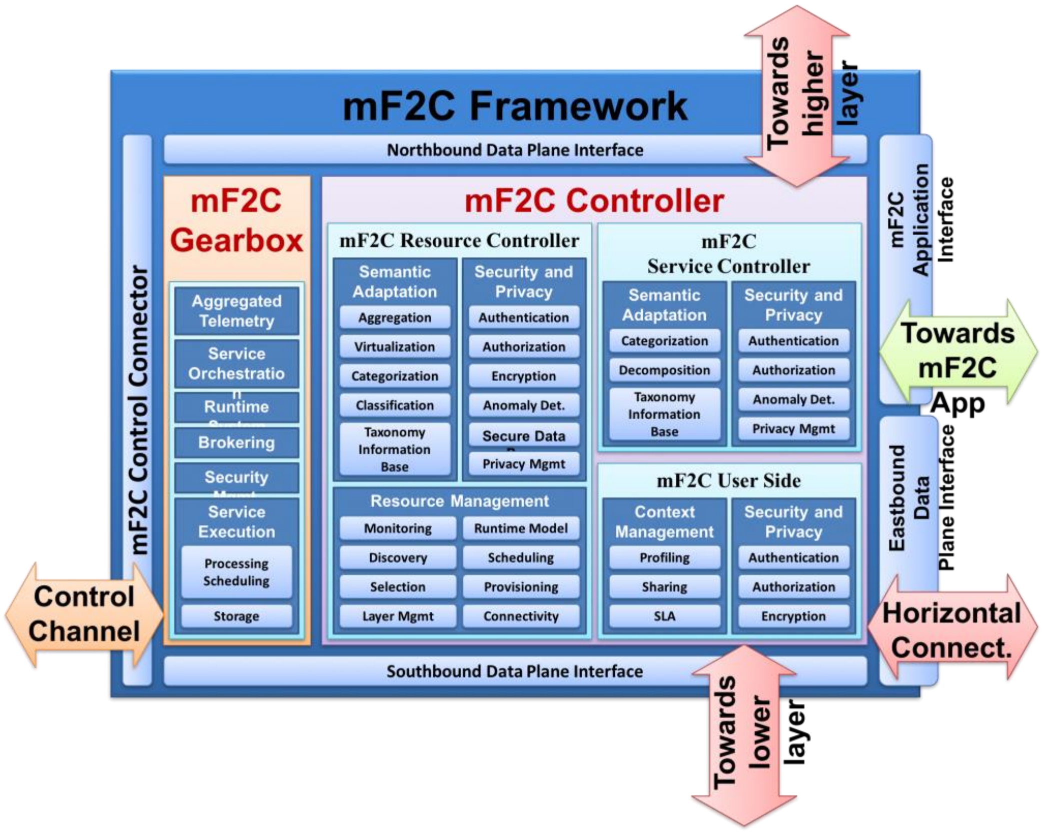


Figure 19: mF2C architecture

This consists of two main blocks: the mF2C controller, and the mF2C gearbox. The controller is responsible for the operational tasks, such as resource management (discovery, monitoring), run-time management (provisioning, scheduling), service management (categorization, decomposition) and user management (profiling, SLA), together with the corresponding security and privacy requirements. The gearbox performs higher level tasks, such as service orchestration, brokering, and telemetry tasks, among others.

The main goals for this project have been defined as follows:

- Defining a global management framework for the F2C ecosystem
- Developing and security and privacy framework
- Developing novel technologies for service execution

The mF2C project has been an ambitious project for the time being and set several fundamental concepts and technologies in the scope of coordinated fog-to-cloud management; however, most of the core components finally run basic control algorithms and applications have to be designed ad hoc. So there is still a huge room for improvement, especially in the scope of intelligent management systems and transparent interfaces.

10.2.2 RECAP

The Reliable Capacity Provisioning and Enhanced Remediation for Distributed Cloud Applications (RECAP) is an EU H2020 research and innovation program funded research project (2017-2019) to research on novel simulation and optimization methodologies for efficient cloud/edge services provisioning, to leverage the resource management automation and optimisation potential.

Building on machine learning, optimisation and simulation techniques, the RECAP project aims to advance the state of the art in the following areas:

- The modeling of complex cloud applications and infrastructures, developing application behaviour models for distributed cloud applications
- The collection of run-time data traces to support orchestration systems for distributed heterogeneous data center infrastructures
- The simulation of large scale cloud/edge/fog scenarios to support orchestration decisions, and detailed simulation models to obtain accurate measurements of storage, file systems, and networks.
- The optimization of data center infrastructure, including scheduling systems, decentralized monitoring and load balancing systems

The RECAP project has provided an advance in the modeling, simulation and optimization of resource management from the collection of complex execution traces in the context of cloud/edge computing: however, this is not a complete framework but a collection of research tools and technologies. Critical aspects such as system control and integration, resources management and system security should be designed and integrated to have a comprehensive running platform.

10.2.3 RAINBOW

The Open, Trusted Fog Computing Platform Facilitating the Deployment, Orchestration and Management of Scalable, Heterogeneous and Secure IoT Services and Cross-Cloud Apps (RAINBOW) is an EU H2020 research and innovation program funded research project (2020-2022) which develops an open and secured fog computing platform for the management of extensible, diverse and safe IoT services and cross-cloud applications.

RAINBOW is a novel platform that simplifies the deployment and management of scalable, heterogeneous and secure IoT services. It has been designed to enable users to remotely control the infrastructure that is running, potentially, on hundreds of edge devices, thousands of fog nodes (for instance, in a factory building or drones flying in the sky), or millions of vehicles traveling in a certain area.

This platform provides deployment, orchestration, network and data management services for scalable and secure edge applications. The RAINBOW architecture is as follows:

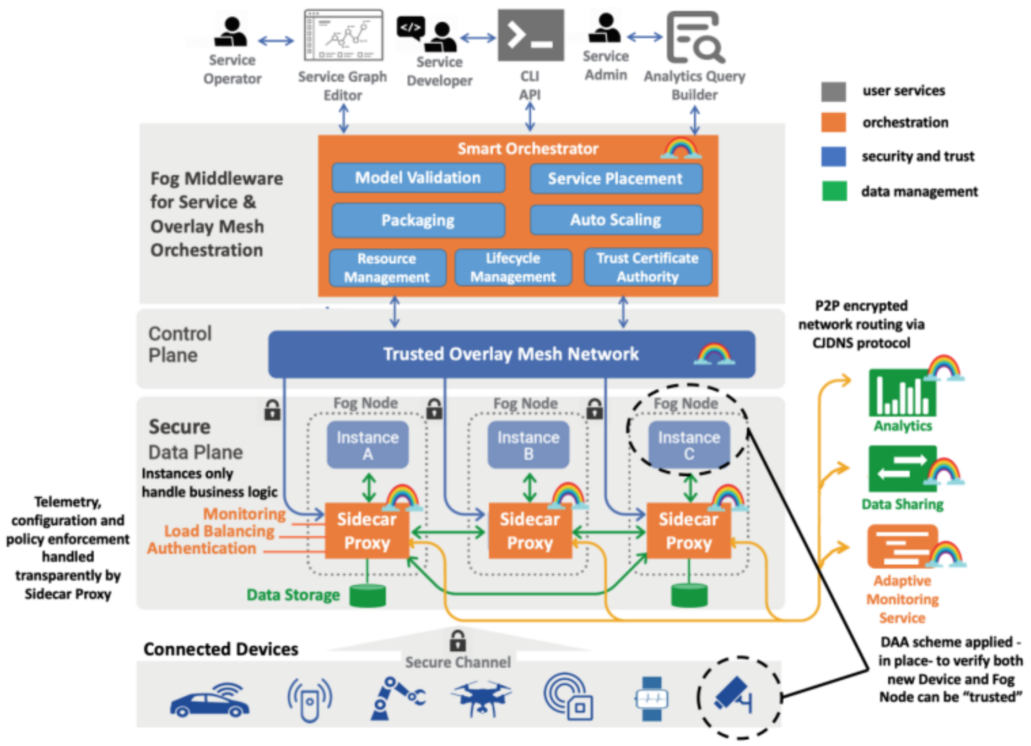


Figure 20: RAINBOW architecture

The main components are the following:

- A Dashboard and toolset for users to enable the description of application topologies and QoS requirements
- The Fog Middleware, with services for resource management and service orchestration, as well as adaptive monitoring and decentralized analysis
- A Sidecar Proxy to provide an execution environment embedded alongside service instances able to properly and efficiently manage both fog node resources and high volumes of data

The RAINBOW project is one step ahead in the design of cloud/fog management platforms and has contributed significant benefits in the area of industry, still coordinated decentralized intelligent solutions and high-level analytic techniques are necessary for effective systems, as well as transparent and portable interfaces for services deployment are required.

10.2.4 Enarx: Trust No One, Run Everywhere

Enarx is an open-source project aimed at facilitating the deployment of workloads to a variety of trusted execution environments (TEE) in the cloud, and ensuring the application workload is as secure as possible.

TEE is a technological approach to run applications within a set of memory pages which are encrypted by the host CPU in such a way that even the owner of the host system is supposed to be unable to peer into or modify the running processes in the TEE instance. Enarx is the framework for running applications in TEE instances facilitating users' programming and management.

Enarx is a security-related system that aims to make it simple to deploy workloads of different TEEs in the cloud, and to provide confidence that the application workload is as secure as possible.

10.2.5 Brain-IoT

The model-Based fRamework for dependable sensing and Actuation in INtelligent decentralized IoT systems (Brain-IoT) is an EU H2020 research and innovation program funded research project (2018-2021) and focuses on complex scenarios where actuation and control are cooperatively supported by populations of IoT systems.

Brain-IoT [61] is a framework for smart and autonomous devices located at the heterogeneous IoT layer. The developers of the project focus on advancements in deployment, assembly and orchestration, as well as management of the system. By combining distributed IoT modules, IoT platforms and IoT services together, IoT architectures can be built. Functionalities for this are available in marketplaces and can be accessed by open APIs.

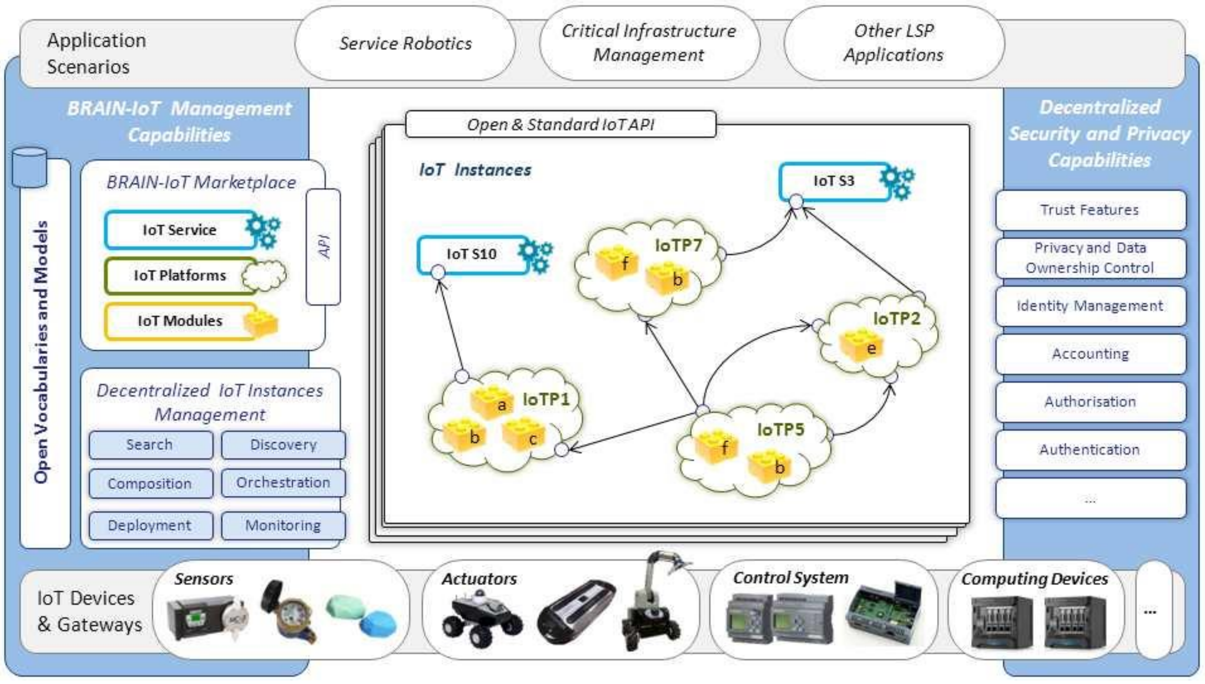


Figure 21: Brain-IoT architecture.

The bottom of the picture displays all physical world IoT devices, with sensing or actuating capabilities. The marketplace provides:

- IoT services: third party services for data storage, data statistics and analytics
- IoT platform: instances of the IoT platforms, they can be configured dynamically
- IoT modules: enabling functionalities for specific IoT platforms

These can meet specified requirements for the IoT applications. A main focus here is the implementation of smart applications of the IoT modules. Brain IoT is developing a library of these IoT modules to enable better context-awareness, real-time data analysis and control solutions.

11 Conclusions

The deliverable D2.1 "ICOS Ecosystem: Technologies, requirements and state of the art" is the first technical report delivered by the ICOS consortium. It presents the results of the work carried out by the activities T2.1 - Ecosystem Identification: Baseline Technologies; T2.2 - Compute Continuum Requirements Definition and T2.3 - AI, Data Management and Trust/Security Requirements during the first six months of the project.

The document introduces the ICOS motivation and challenges to define the ICOS concepts and artifacts. These definitions are indeed still preliminary and are intended just to support the requirements elicitation activity.

The document also identified ICOS stakeholders and their roles in the ICOS ecosystem. Those stakeholders are involved in a series (eight) of User Stories that explain the method by which the ICOS will manage and control Cloud Continuum as well as the key drivers and the incentive that set the context in which solutions and services of the platform should be offered.

Furthermore, D2.1 has provided a first high level description of the four Use Cases will assess and validate ICOS technologies:

- UC1: Agriculture Operational Robotic Platform;
- UC2: Railway Structural Alert Monitoring system;
- UC3: In-car Advanced Infotainment and Multimedia Management system;
- UC4: Energy Management and Decision Support system)

ICOS Requirements (Functional and Non-Functional) Elicitation has been as the result of analysis and a series of activities carried out within the different actors and stakeholders of the project's value chain.

This analysis has been carried out from five different points of view, which are considered to be the main themes of the project, in order to produce the functional requirements:

- Continuum Creation:
- Continuum Management:
- Data Resiliency and Transformation:
- Smart Security and Trust:
- Operability Serviceability (GUI/CLI)

In order to provide a set of requirements, the User Stories and the validation Use Cases were studied and the results from mentioned analysis are detailed in a set of tables. This output is produced by following the MoSCoW method in order to reduce the requirements elicitation extension.

Finally deliverable tracked scientific, technology and business trends in the area of Cloud, Edge and IoT computing that are relevant to the ICOS project in the field of the service and infrastructure management, data processing and management, related research project. This work will give a generic overview of all technological trends of which awareness is necessary for the project, and for the proposed area of research at large.

This is the first version of the deliverable (v1), aligned to iteration 1 (IT-1) of the project. A second version (M20), aligned to iteration 2 (IT-2), is due to be delivered within the second project period.

12 References

[1] Docker Engine, https://www.docker.com/products/container-runtime/, retrieved 03/03/2023

[2] Submariner, https://submariner.io/, retrieved 03/03/2023

[3] Kubernetes Multi-Cluster Services, https://github.com/kubernetes/enhancements/tree/master/keps/sig-multicluster/1645-multi-cluster-services-api, retrieved 03/03/2023

[4] Genie home page, https://netflix.github.io/genie/, retrieved 03/03/2023

[5] About Genie, https://netflix.github.io/genie/about/, retrieved 03/03/2023

[6] Nuvla, https://docs.nuvla.io/ and https://sixsq.com/platform retrieved 03/03/2023

[7] HPE GreenLake, https://www.hpe.com/us/en/solutions/edge-to-cloud.html and https://www.wwt.com/article/what-is-hpe-greenlake, retrieved 03/03/2023

[8] What is AWS IoT Greengrass, https://docs.aws.amazon.com/greengrass/v2/developerguide/what-is-iot-greengrass.html and https://docs.aws.amazon.com/greengrass/v2/developerguide/how-it-works.html, retrieved 03/03/2023

[9] Apache OpenWhisk, https://openwhisk.apache.org/, retrieved 03/03/2023

[10] OpenFaas, https://docs.openfaas.com/, retrieved 03/03/2023

[11] Machine Learning Operations (MLOps): Overview, Definition, and Architecture https://arxiv.org/abs/2205.02302), retrieved 06/03/2023

[12] Huggin Face Hub, https://huggingface.co/docs/hub/index, retrieved 06/03/2023

[13] Data Version Control (DVC), https://dvc.org/, retrieved 06/03/2023

[14] ML Metadata, https://github.com/google/ml-metadata, retrieved 06/03/2023

[15] Mercurial, https://www.mercurial-scm.org/, retrieved 06/03/2023

[16] Modelzoo, https://modelzoo.co /, retrieved 06/03/2023

[17] SingularityNet, https://singularitynet.io/, retrieved 06/03/2023

[18] AWS Marketplace, https://aws.amazon.com/marketplace, retrieved 06/03/2023

[19] Genessai, https://www.genesisai.io/, retrieved 06/03/2023

[20] Rocklin, Matthew. "Dask: Parallel computation with blocked algorithms and task scheduling." Proceedings of the 14th python in science conference. Vol. 130. Austin, TX: SciPy, 2015. doi: 10.25080/Majora-7b98e3ed-013

[21] Zaharia, Matei, et al. "Apache spark: a unified engine for big data processing." Communications of the ACM 59.11 (2016): 56-65. doi: 10.1145/2934664

[22] Lordan, Francesc, et al. "Servicess: An interoperable programming framework for the cloud." Journal of grid computing 12 (2014): 67-91. doi: 10.1007/s10723-013-9272-5

[23] Babuji, Yadu, et al. "Parsl: Pervasive parallel programming in python." Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing. 2019. doi: 10.1145/3307681.3325400

[24] Zenoh-flow, https://zenoh.io/blog/2023-02-10-zenoh-flow/ , retrieved 06/03/2023

[25] Ramon-Cortes, Cristian, et al. "A Programming Model for Hybrid Workflows: combining Task-based Workflows and Dataflows all-in-one." Future Generation Computer Systems 113 (2020): 281-297. doi: 10.1016/j.future.2020.07.007

[26] Lordan, Francesc, Daniele Lezzi, and Rosa M. Badia. "Colony: Parallel Functions as a Service on the Cloud-Edge Continuum." Euro-Par 2021: Parallel Processing: 27th International Conference on Parallel and Distributed Computing, Lisbon, Portugal, September 1–3, 2021, Proceedings 27. Springer International Publishing, 2021. doi: 10.1007/978-3-030-85665-6_17

[27] Chard, Ryan, et al. "Funcx: A federated function serving fabric for science." Proceedings of the 29th International symposium on high-performance parallel and distributed computing. 2020. doi: 10.1145/3369583.3392683

[28] Das, Sumit, et al. "Applications of artificial intelligence in machine learning: review and prospect." International Journal of Computer Applications 115.9 (2015). doi: 10.5120/20182-2402

[29] Brecko, Alexander, et al. "Federated Learning for Edge Computing: A Survey." Applied Sciences 12.18 (2022): 9124. doi: 10.3390/app12189124

[30] K. H. Prasad, T. A. Faruquie, S. Joshi, S. Chaturvedi, L. V. Subramaniam, and M. Mohania, “Data Cleansing Techniques for Large Enterprise Datasets,” in 2011 Annual SRII Global Conference, Mar. 2011, pp. 135–144. doi: 10.1109/SRII.2011.26.

[31] A. M. Jonathan I. Maletic, “Data Cleansing: Beyond Integrity Analysis,” p. 10, 2000.

[32] X. Wu, X. Gao, W. Zhang, R. Luo, and J. Wang, “Learning over categorical data using counting features: with an application on click-through rate estimation,” in Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, New York, NY, USA, Aug. 2019, pp. 1–9. doi: 10.1145/3326937.3341260.

[33] A. Kratsios and C. Hyndman, “NEU: A Meta-Algorithm for Universal UAP-Invariant Feature Representation,” J. Mach. Learn. Res., vol. 22, no. 92, pp. 1–51, 2021.

[34] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” J. Mach. Learn. Res., vol. 3, no. Mar, pp. 1157–1182, 2003.

[35] S. Piramuthu, H. Ragavan, and M. J. Shaw, “Using Feature Construction to Improve the Performance of Neural Networks,” Manag. Sci., vol. 44, no. 3, pp. 416–430, Mar. 1998, doi: 10.1287/mnsc.44.3.416.

[36] A. Nayak, A. Kanive, N. Chandavekar, and B. Ramasamy, “Survey on Pre-Processing Techniques for Text Mining,” Int. J. Eng. Comput. Sci., vol. 5, pp. 2319–7242, Jun. 2016, doi: 10.18535/ijecs/v5i6.25.

[37] Y. A. Ibrahim, J. C. Odiketa, and T. S. Ibiyemi, “PREPROCESSING TECHNIQUE IN AUTOMATIC SPEECH RECOGNITION FOR HUMAN COMPUTER INTERACTION: AN OVERVIEW,” p. 6, 2017.

[38] Apache Kafka Event-Driven Workflow Orchestration, https://www.projectpro.io/article/apache-kafka-architecture-/442, retrieved 06/03/2023

[39] Eclipse Zenoh – What is a Zenoh?, https://zenoh.io/docs/overview/what-is-zenoh/, retrieved 06/03/2023

[40] R. Tanaka et al., "Automating Edge-to-cloud Workflows for Science: Traversing the Edge-to-cloud Continuum with Pegasus," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 826-833, doi: 10.1109/CCGrid54584.2022.00098.

[41] Suyi Li, Luping Wang, Wei Wang, Yinghao Yu, and Bo Li. 2021. George: Learning to Place Long-Lived Containers in Large Clusters with Operation Constraints. In Proceedings of the ACM Symposium on Cloud Computing (SoCC '21). Association for Computing Machinery, New York, NY, USA, 258–272. https://doi.org/10.1145/3472883.3486971

[42] Vivek M. Bhasi, Jashwant Raj Gunasekaran, Prashanth Thinakaran, Cyan Subhra Mishra, Mahmut Taylan Kandemir, and Chita Das. 2021. Kraken: Adaptive Container Provisioning for Deploying Dynamic DAGs in Serverless Platforms. In Proceedings of the ACM Symposium on Cloud Computing (SoCC '21). Association for Computing Machinery, New York, NY, USA, 153–167. https://doi.org/10.1145/3472883.3486992

[43] P. Liu, G. Bravo-Rocca, J. Guitart, A. Dholakia, D. Ellison and M. Hodak, "Scanflow-K8s: Agent-based Framework for Autonomic Management and Supervision of ML Workflows in Kubernetes Clusters," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 376-385, doi: 10.1109/CCGrid54584.2022.00047.

[44] M. Han and W. Baek, "HERTI: A Reinforcement Learning-Augmented System for Efficient Real-Time Inference on Heterogeneous Embedded Systems," 2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Atlanta, GA, USA, 2021, pp. 90-102, doi: 10.1109/PACT52795.2021.00014.

[45] J. Xu, B. Palanisamy, H. Ludwig and Q. Wang, "Zenith: Utility-Aware Resource Allocation for Edge Computing," 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 2017, pp. 47-54, doi: 10.1109/IEEE.EDGE.2017.15.

[46] M. S. Aslanpour, A. N. Toosi, M. A. Cheema and R. Gaire, "Energy-Aware Resource Scheduling for Serverless Edge Computing," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 190-199, doi: 10.1109/CCGrid54584.2022.00028.

[47] S. Nastic et al., "Polaris Scheduler: Edge Sensitive and SLO Aware Workload Scheduling in Cloud-Edge-IoT Clusters," 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 2021, pp. 206-216, doi: 10.1109/CLOUD53861.2021.00034.

[48] Sing, R.; Bhoi, S.K.; Panigrahi, N.; Sahoo, K.S.; Bilal, M.; Shah, S.C. EMCS: An Energy-Efficient Makespan Cost-Aware Scheduling Algorithm Using Evolutionary Learning Approach for Cloud-Fog-Based IoT Applications. Sustainability 2022, 14, 15096. https://doi.org/10.3390/su142215096

[49] Mutichiro B, Tran M-N, Kim Y-H. QoS-Based Service-Time Scheduling in the IoT-Edge Cloud. Sensors. 2021; 21(17):5797. https://doi.org/10.3390/s21175797

[50] A. Luckow, K. Rattan and S. Jha, "Pilot-Edge: Distributed Resource Management Along the Edge-to-Cloud Continuum," 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Portland, OR, USA, 2021, pp. 874-878, doi: 10.1109/IPDPSW52791.2021.00130.

[51] J. Zhu, J. Wang, Y. Huang, F. Fang, K. Navaie and Z. Ding, "Resource Allocation for NOMA MEC Offloading," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013239.

[52] Z. Gan, R. Lin and H. Zou, "A Multi-Agent Deep Reinforcement Learning Approach for Computation Offloading in 5G Mobile Edge Computing," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 645-648, doi: 10.1109/CCGrid54584.2022.00074

[53] X. Zhang, L. Cui and F. P. Tso, "pSFC: Fine-grained Composition of Service Function Chains in the Programmable Data Plane," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 484-493, doi: 10.1109/CCGrid54584.2022.00058.

[54] N. Ferry, R. Dautov and H. Song, "Towards a Model-Based Serverless Platform for the Cloud-Edge-IoT Continuum," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 851-858, doi: 10.1109/CCGrid54584.2022.00101.

[55] Fedor Smirnov, Behnaz Pourmohseni, and Thomas Fahringer. 2021. Apollo: Modular and Distributed Runtime System for Serverless Function Compositions on Cloud, Edge, and IoT Resources. In Proceedings of the 1st Workshop on High Performance Serverless Computing (HiPS '21). Association for Computing Machinery, New York, NY, USA, 5–8. https://doi.org/10.1145/3452413.3464793

[56] Chengliang Zhang, Junzhe Xia, Baichen Yang, Huancheng Puyang, Wei Wang, Ruichuan Chen, Istemi Ekin Akkus, Paarijaat Aditya, and Feng Yan. 2021. Citadel: Protecting Data Privacy and Model Confidentiality for Collaborative Learning. In Proceedings of the ACM Symposium on Cloud Computing (SoCC '21). Association for Computing Machinery, New York, NY, USA, 546–561. https://doi.org/10.1145/3472883.3486998

[57] R. Saiz-Laudo and M. Sánchez-Artigas, "Egeon: Software-Defined Data Protection for Object Storage," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 99-108, doi: 10.1109/CCGrid54584.2022.00019.

[58] J. Aljabri, A. L. Michala and J. Singer, "ELSA: A Keyword-based Searchable Encryption for Cloud-edge assisted Industrial Internet of Things," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 259-268, doi: 10.1109/CCGrid54584.2022.00035.

[59] Yuepeng Li, Deze Zeng, Lin Gu, Quan Chen, Song Guo, Albert Zomaya, and Minyi Guo. 2021. Lasagna: Accelerating Secure Deep Learning Inference in SGX-enabled Edge Cloud. In Proceedings of the ACM Symposium on Cloud Computing (SoCC '21). Association for Computing Machinery, New York, NY, USA, 533–545. https://doi.org/10.1145/3472883.3486988

[60] V. Mygdalis et al., "OTE: Optimal Trustworthy EdgeAI solutions for smart cities," 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 2022, pp. 842-850, doi: 10.1109/CCGrid54584.2022.00100.

[61] BRAIN-Iot project concept https://www.brain-iot.eu/about/concept/, retrieved 06/03/2023

[62] AWS Cloud 9 overview, https://aws.amazon.com/cloud9/?nc1=h_ls, retrieved 09/02/2023

[63] AWS CloudShell overview, https://aws.amazon.com/cloudshell/

[64] AWS Well-Architected Framework concepts: AWS Cloud Trail, https://wa.aws.amazon.com/wellarchitected/2020-07-02T19-33-23/wat.concept.cloudtrail.en.html

[65] Kubernetes Platform. (2015, July 21). Retrieved 2022, from https://kubernetes.io/docs/concepts/overview/

[66] K3s - Lightweight Kubernetes. (2019). Retrieved 2022, from https://docs.k3s.io/architecture

[67] Rancher. (2022). Rancher Kubernetes Management. Retrieved from Rancher Kubernetes Management: https://rancher.com/docs/

[68] Knative. (2022). Knative. Retrieved from Knative: https://knative.dev/docs/concepts/#knative-serving

[69] Open Cluster Management, https://open-cluster-management.io/concepts/architecture/

[70] Qiang Yang , Yang Liu , Yong Cheng , Yan Kang , Tianjian Chen , Han Yu. “Federated Learning”, https://link.springer.com/book/10.1007/978-3-031-01585-4, retrieved 06/03/2023

[71] X. Liu et al., “UniFed: A Benchmark for Federated Learning Frameworks.” arXiv, Oct. 20, 2022. doi: 10.48550/arXiv.2207.10308.

[72] C. He et al., “FedML: A Research Library and Benchmark for Federated Machine Learning.” arXiv, Nov. 08, 2020. doi: 10.48550/arXiv.2007.13518.

[73] M. H. Garcia, A. Manoel, D. M. Diaz, F. Mireshghallah, R. Sim, and D. Dimitriadis, “FLUTE: A Scalable, Extensible Framework for High-Performance Federated Learning Simulations.” arXiv, Nov. 14, 2022. doi: 10.48550/arXiv.2203.13789.

[74] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, “CrypTen: Secure Multi-Party Computation Meets Machine Learning.” arXiv, Sep. 15, 2022. doi: 10.48550/arXiv.2109.00984.

[75] Y. C. Q. Li, “Welcome to FedTree’s documentation! — FedTree documentation.” https://github.com/Xtra-Computing/FedTree/ blob/main/FedTree_draft_paper.pdf (accessed Dec. 05, 2022).

[76] Top 7 Open-Source Frameworks for Federated Learning, published on 09/09/2022, https://www.apheris.com/resources/blog/top-7-open-source-frameworks-for-federated-learning, retrieved on 07/03/2023

[77] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Apr. 2017, pp. 1273–1282. Accessed: Dec. 05, 2022. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[78] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated Optimization in Heterogeneous Networks,” Proc. Mach. Learn. Syst., vol. 2, pp. 429–450, Mar. 2020.

[79] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, “Federated Learning with Personalization Layers.” arXiv, Dec. 02, 2019. doi: 10.48550/arXiv.1912.00818.

[80] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, “FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation,” Nat. Commun., vol. 13, no. 1, p. 3091, Jun. 2022, doi: 10.1038/s41467-022-30714-9.

[81] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” in Advances in Neural Information Processing Systems, 2017, vol. 30. Accessed: Dec. 05, 2022. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[82] K. ZHANG, C. Yang, X. Li, L. Sun, and S. M. Yiu, “Subgraph Federated Learning with Missing Neighbor Generation,” in Advances in Neural Information Processing Systems, 2021, vol. 34, pp. 6671–6682. https://proceedings.neurips.cc/paper/2021/hash/34adeb8e3242824038aa65460a47c29e-Abstract.html, retrieved 05/12/2022

[83] S. Reddi et al., “Adaptive Federated Optimization.” arXiv, Sep. 08, 2021. doi: 10.48550/arXiv.2003.00295.

[84] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, “SplitFed: When Federated Learning Meets Split Learning,” Proc. AAAI Conf. Artif. Intell., vol. 36, no. 8, Art. no. 8, Jun. 2022, doi: 10.1609/aaai.v36i8.20825.

[85] Z. Zhang, N. He, D. Li, H. Gao, T. Gao, and C. Zhou, “Federated transfer learning for disaster classification in social computing networks,” J. Saf. Sci. Resil., vol. 3, no. 1, pp. 15–23, Mar. 2022, doi: 10.1016/j.jnlssr.2021.10.007.

[86] L. Tu, X. Ouyang, J. Zhou, Y. He, and G. Xing, “FedDL: Federated Learning via Dynamic Layer Sharing for Human Activity Recognition,” in Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, New York, NY, USA, Nov. 2021, pp. 15–28. doi: 10.1145/3485730.3485946.

[87] H. Zhu and Y. Jin, “Multi-objective Evolutionary Federated Learning.” arXiv, Jun. 08, 2019. doi: 10.48550/arXiv.1812.07478.

[88] West, Jeremy, “A Theoretical Foundation for Inductive Transfer,” Aug. 01, 2007. https://web.archive.org/web/20070801120743/http://cpms.byu.edu/springresearch/abstract-entry?id=861, retrieved 13/12/2022.

[89] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[90] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both Weights and Connections for Efficient Neural Network,” in Advances in Neural Information Processing Systems, 2015, vol. 28. https://papers.nips.cc/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html, retrieved 05/12/2022.

[91] [19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[92] [20] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network.” arXiv, Mar. 09, 2015. http://arxiv.org/abs/1503.02531, retrieved 05/12/2022.

[93] Gou, J., Yu, B., Maybank, S.J. and Tao, D., 2021. Knowledge distillation: A survey. International Journal of Computer Vision, 129(6), pp.1789-1819. https://link.springer.com/article/10.1007/s11263-021-01453-z

[94] [21] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning Efficient Object Detection Models with Knowledge Distillation,” in Advances in Neural Information Processing Systems, 2017, vol. 30. https://proceedings.neurips.cc/paper/2017/hash/e1e32e235eee1f970470a3a6658dfdd5-Abstract.html, retrieved 05/12/2022.

[95] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013, doi: 10.1109/TPAMI.2013.50.

[96] J. Yim, D. Joo, J. Bae, and J. Kim, “A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, Jul. 2017, pp. 7130–7138. doi: 10.1109/CVPR.2017.754.

[97] G. Fang, J. Song, C. Shen, X. Wang, D. Chen, and M. Song, “Data-Free Adversarial Distillation.” arXiv, Mar. 02, 2020. doi: 10.48550/arXiv.1912.11006.

[98] Cheol-Ho Hong, Blesson VargheseResource. “Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms”. https://arxiv.org/pdf/1810.00305.pdf, retrieved 07/03/2023.

[99] C. L. Hoang T. Dinh, “A survey of mobile cloud computing: architecture, applications, and approaches,” 11 Oct. 2011, doi: 10.1002/wcm.1203.

[100] [26] X. Chen, L. Jiao, W. Li, and X. Fu, “Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing,” IEEEACM Trans. Netw., vol. 24, no. 5, pp. 2795–2808, Oct. 2016, doi: 10.1109/TNET.2015.2487344.

[101] I. Stojmenovic, “Fog computing: A cloud to the ground support for smart things and machine-to-machine networks,” 2014 Australas. Telecommun. Netw. Appl. Conf. ATNAC, pp. 117–122, Nov. 2014, doi: 10.1109/ATNAC.2014.7020884.

[102] Lordan, F., Lezzi, D., Badia, R.M. (2021). Colony: Parallel Functions as a Service on the Cloud-Edge Continuum. In: Sousa, L., Roma, N., Tomás, P. (eds) Euro-Par 2021: Parallel Processing. Euro-Par 2021. Lecture Notes in Computer Science(), vol 12820. Springer, Cham. https://doi.org/10.1007/978-3-030-85665-6_17

[103] Z. Li et al., "𝑓uncX: Federated Function as a Service for Science," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 12, pp. 4948-4963, 1 Dec. 2022, doi: 10.1109/TPDS.2022.3208767.

[104] Abadi, Martín; Barham, Paul; Chen, Jianmin; Chen, Zhifeng; Davis, Andy; Dean, Jeffrey; Devin, Matthieu; Ghemawat, Sanjay; Irving, Geoffrey; Isard, Michael; Kudlur, Manjunath; Levenberg, Josh; Monga, Rajat; Moore, Sherry; Murray, Derek G.; Steiner, Benoit; Tucker, Paul; Vasudevan, Vijay; Warden, Pete; Wicke, Martin; Yu, Yuan; Zheng, Xiaoqiang (2016). TensorFlow: A System for Large-Scale Machine Learning (PDF). Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16). arXiv:1605.08695

[105] Ketkar, Nikhil (2017). "Introduction to PyTorch". Deep Learning with Python. Apress, Berkeley, CA. pp. 195–208. doi:10.1007/978-1-4842-2766-4_12. ISBN 9781484227657

[106] Sing, R.; Bhoi, S.K.; Panigrahi, N.; Sahoo, K.S.; Bilal, M.; Shah, S.C. EMCS: An EnergyEfficient Makespan Cost-Aware Scheduling Algorithm Using Evolutionary Learning Approach for Cloud-Fog-Based IoT Applications. Sustainability 2022, 14, 15096. https://doi.org/10.3390/su142215096

[107] Ali, Syed & Amsari, Manzoor & Alam, Mansaf. (2020). Resource Management Techniques for Cloud-Based IoT Environment. pp. 9-11. doi:10.1007/978-3-030-37468-6_4.

[108] OpenEuler (2020), https://docs.openeuler.org/en/, Retrieved 06/03/2023.

[109] OpenFaaS, https://www.openfaas.com/, Retrieved 06/03/2023.

[110] A. M. Jonathan I. Maletic, “Data Cleansing: Beyond Integrity Analysis,” p. 10, 2000.

[111] X. Wu, X. Gao, W. Zhang, R. Luo, and J. Wang, “Learning over categorical data using counting features: with an application on click-through rate estimation,” in Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, New York, NY, USA, Aug. 2019, pp. 1–9. doi: 10.1145/3326937.3341260.

[112] A. Kratsios and C. Hyndman, “NEU: A Meta-Algorithm for Universal UAP-Invariant Feature Representation,” J. Mach. Learn. Res., vol. 22, no. 92, pp. 1–51, 2021

[113] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” J. Mach. Learn. Res., vol. 3, no. Mar, pp. 1157–1182, 2003.

[114] S. Piramuthu, H. Ragavan, and M. J. Shaw, “Using Feature Construction to Improve the Performance of Neural Networks,” Manag. Sci., vol. 44, no. 3, pp. 416–430, Mar. 1998, doi: 10.1287/mnsc.44.3.416.

[115] A. Nayak, A. Kanive, N. Chandavekar, and B. Ramasamy, “Survey on Pre-Processing Techniques for Text Mining,” Int. J. Eng. Comput. Sci., vol. 5, pp. 2319–7242, Jun. 2016, doi: 10.18535/ijecs/v5i6.25.

[116] Y. A. Ibrahim, J. C. Odiketa, and T. S. Ibiyemi, “PREPROCESSING TECHNIQUE IN AUTOMATIC SPEECH RECOGNITION FOR HUMAN COMPUTER INTERACTION: AN OVERVIEW,” p. 6, 2017.

Annexes

1 Continuum management technologies

1.1 Cloud Based Operating Systems

1.1.1 Openstack

Openstack is a cloud operating system that can integrate various computing and networking technologies via a common platform that provides the necessary APIs for user authentication, control, and operation. In most of the cases it is deployed as infrastructure-as-a-service (IaaS) in both public and private clouds, where virtual servers and in general other resources are made available to the end users. OpenStack can be viewed as a cost-effective extension of the existing public cloud infrastructure, giving to organizations and individuals the ability to optimize their cloud maintenance costs and service providers to build an infrastructure that can be easily expandable and interoperable with other operating environments.

The basic idea of OpenStack as illustrated in Figure 22: Openstack supported servicesbelow. It is based on the decomposition of the system into multiple independent services and functionalities, where the appropriate tools and APIs are inserted per case. These include among others shared services, software defined networking, hardware lifecycle, compute and storage, workload provision, application lifecycle, orchestration, and web front end.

Moreover, OpenStack supports application interoperability, since the same application can be deployed on a variety of heterogeneous OpenStack clouds, compatibility with other versions, cross-project dependencies, since multiple code segments might exist in various locations, as well as partitioning of end services to avoid overloading.

However, compatibility with other cloud APIs is limited.

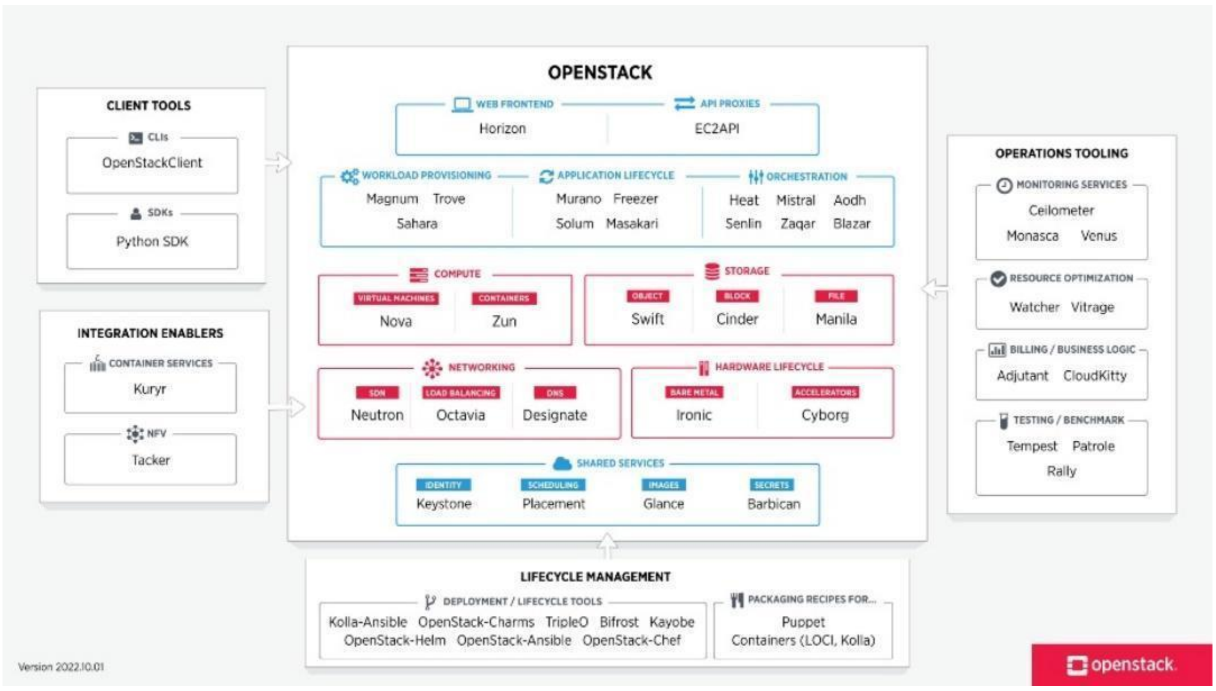


Figure 22: Openstack supported services

OpenStack supports a modular architecture, where various components with specific functionalities are integrated. The most important ones include the following (coded names have been adopted):

- **Keystone** provides identity services for OpenStack, controlling user authentication.
- **Ceilometer** is a network telemetry functionality which allows real time monitoring of the users connected to the cloud, in order to enforce appropriate billing mechanisms. However, resource utilization monitoring is supported as well.
- **Neutron** component, is responsible for ensuring network connectivity as a service to end users, including among others software defined networking and virtual machine support.
- **Horizon** is the OpenStack's GUI, where application developers can have access to all related functionalities via APIs.
- **Nova** is responsible for computational tasks based on virtual machines concept.
- **Heat** supports service orchestration that can span across multiple environments.

In the same context, the concept of hardware virtualization is supported since vendor-independent APIs are provided for the end user to have control over various hardware components. Moreover, continuous scaling is supported as well, where various workloads can be managed without the need for architectural redesigns.

1.1.2 AWS

Amazon Web Services (AWS) offer a variety of functionalities related to the proper management of cloud infrastructure, such as computing, storage, data analytics, security, and enterprise applications. The most important tools are listed below:

- **Cloud Financial Management**, which includes functionalities such as cost profiler, where the ability to track the consumption of shared AWS is provided, billing contractor, cost explorer, cost, and usage reports, etc. Therefore, end users can be benefited since they can be informed on the exact utility costs.
- **Amazon Managed Blockchain** which allows the creation of blockchains via open-source tools Hyperledger Fabric and Ethereum. In general, blockchain is a significant emerging technology that can leverage secure authentication and privacy protection in cloud environments.
- **Analytics**, where the Amazon CloudSearch tool supports search solutions over websites or applications. Therefore, context search over various sources is supported.
- **Containers**, which supports tools related to containers initialization and management, as well as Kubernetes.
- **AWS Cloud9** [62] is a cloud-based integrated development environment (IDE) that allows programmers to run and debug their code with just a browser.
- **AWS CloudShell** [63] that allows secure interactions with AWS resources.
- **Amazon CloudWatch** is a monitoring and management service particularly designed for developers, system operators, and IT managers. CloudWatch provides end users with data and other related info to monitor applications, understand and respond to system-wide performance changes, and optimize resource utilization. Therefore, a holistic management approach is provided, with a unified view of AWS resources.
- **AWS CloudFormation**, that gives developers the opportunity to create and manage AWS resources
- **AWS CloudTrail** [64] that provides end users with a record tool that tracks all instances that a particular API was requested. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service. Therefore, appropriate billing mechanisms can be enforced according to API utilization.
- **AWS Config**, that gives the opportunity for resource inventory, configuration history, and configuration change notifications to enable security and governance

1.2 Edge Containerization and Orchestration

1.2.1 Kubernetes or K8s

Kubernetes [65] is an open-source multi-cluster orchestration platform that provides a set of operations to manage and automate life-cycle processes such as scaling or deploying multi-cluster applications. Initially the focus was put onto scheduling and dynamic placement but later additional control tools such as pods, labels and replication controller were added. A pod is a minimal deployable object in Kubernetes, it usually represents an instance of running process in a cluster. It contains one or multiple containers that share resources and it is managed as a single entity.

Kubernetes nowadays is the most used platform since it offers a large set of features for cluster management including high availability deployments, load balancing and scalability. These features are the main difference between Kubernetes and a standard Function as a Service (FaaS) Platform.

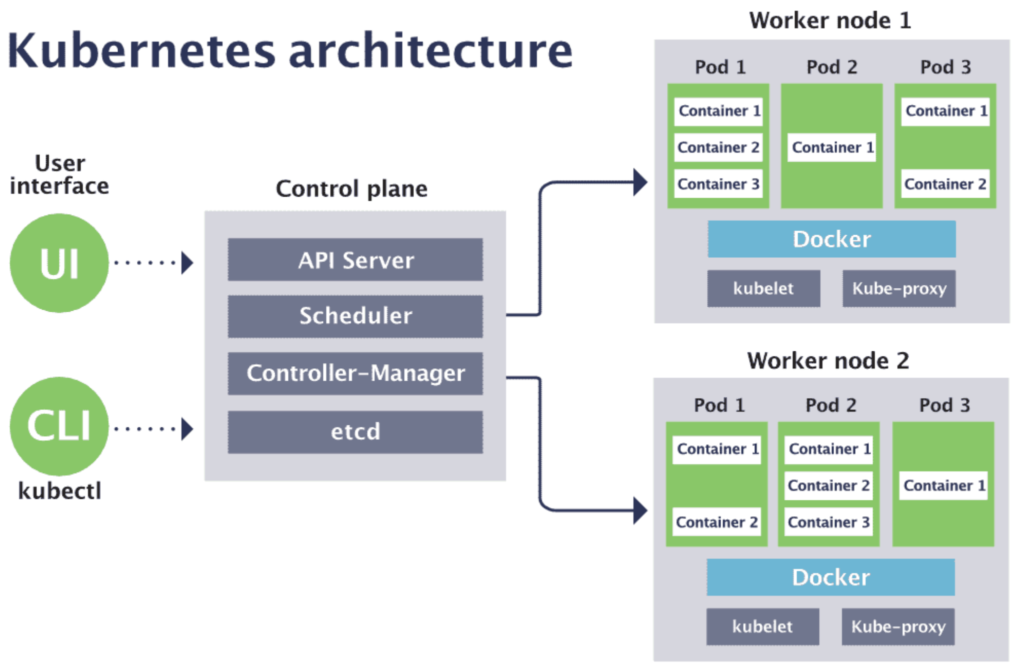


Figure 23: Kubernetes architecture

To understand how Kubernetes provides mentioned features and how this cluster management is accomplished, Kubernetes architecture and its key components are presented in this section.

Kubernetes consists of three basic components:

- **Kubernetes Control Plane**: Key component of Kubernetes in charge of controlling and managing cluster nodes. It maintains the state and configuration data of each cluster node and related infrastructure. To control mentioned cluster nodes Control Plane is constantly connected to compute machines through so called controllers. These controllers are aware of the current state of a node and will ensure to match it with the desired state of the system nodes. In other words, Control Plane will ensure that clusters are run as configured.

  Kubernetes Control Plane itself is composed by several components:

  - **Kubernetes API Server**: API that allow to manage all supported lifecycle operations such as scale or discovery. This API must be accessible from outside the cluster since it is used as a gateway to access and orchestrate clusters as well as all its nodes and services. It also provides authentication service to provide a security layer for mentioned API server.

- **Kubernetes Scheduler**: overwatches and stores the resource usage data for each compute node. This data allows scheduler to determine whether the cluster is in the desired state, or new containers need to be deployed. In case new containers must be deployed, scheduler will also determine where to place them.
- **Kubernetes Controller Manager**: as mentioned before, control plane contains multiple controllers that overwatch and manage state of cluster nodes. Controller Manager is a daemon that observes node state via API server and leverages it towards desired state using these controllers. Kubernetes Controller Manager also performs core lifecycle operations (scaling, healing, etc.).
- **Distributed Key-value System** (ETCD): is an open-source, key-value database that stores data and information about the state of the cluster. ETCD is not a part of Control Plane per se since it can be configured externally, but most of the times is used as part of the Control Plane.

  ETCD stores cluster state data based on the Raft consensus protocol. This protocol uses so called RAFT consensus algorithm to avoid well-known issues derived directly from the distributed consensus problem in the context of multi-replicated state machine. Raft defines three different roles: leader, candidate, and follower, and achieves consensus by electing a leader.

  Within consensus protocol and mentioned roles, ETCD acts as a single source of truth (SSOT) for all Kubernetes cluster components, responding to requests from the control plane and retrieving information about the state of containers, nodes, and pods. As mentioned before, ETCD also provides a storage for container state configuration such as Secrets or ConfigMap.

- **Cluster Nodes**: physical machines that run containerized applications. A cluster node is formed by a set of different components and services, the most significant ones are detailed as follows:
  - Nodes: virtual or physical machine that contains all previously mentioned Kubernetes components necessary for running pods.
  - Pods: smallest deployable units of computing that can be created by Kubernetes. It is usually a group of containers with shared resources and specifications.
  - Container Runtime Engine
  - Kubelet Service: agent that runs on each node of the cluster. It ensures that all containers are running as defined by PodSpecs in the corresponding pod.
  - Kube-proxy Service: network proxy running on each node of the cluster. In charge of network maintenance and provision for pods.

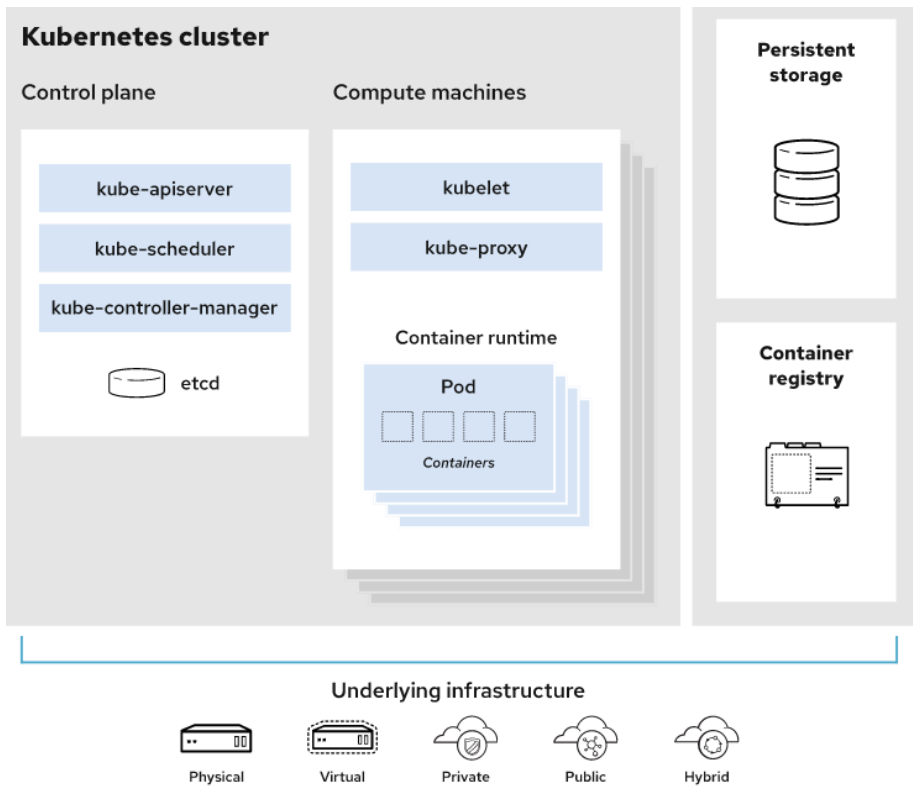


Figure 24: Kubernetes Cluster

1.2.2 Lightweight Kubernetes or K3s

Commonly named K3s [66] is a Kubernetes lightweight distribution developed by Rancher Labs and certified by Cloud Native Computing Foundation as a Kubernetes-compliant distribution. K3s are not functionally different from standard Kubernetes (K8s), but they have some important differences that clearly differentiate them. The main difference is that K3s being a lightweight distribution is much faster in deployments than K8s. This is accomplished by running Kubernetes on bare-metal servers. In previous section Kubelet component of Kubernetes were explained. This agent runs as a container on each node, its purpose is to perform control loops on the containers from a node. For Lightweight Kubernetes this is not true since their Kubelet runs on top of the host machine and uses hosts capabilities to run containers directly.

Other changes or enhancements on K3s over K8s are:

- Packaged as a single binary that is half the size of a standard Kubernetes.
- All features offered by Kubernetes control plane are embedded in a single binary and process.
- Great reduction of external dependencies including network services controllers.
- Operable in Edge Computing or IoT devices.

Disadvantages:

- Can only host workloads that run in a single cloud.
- Reduced performance with big workloads.

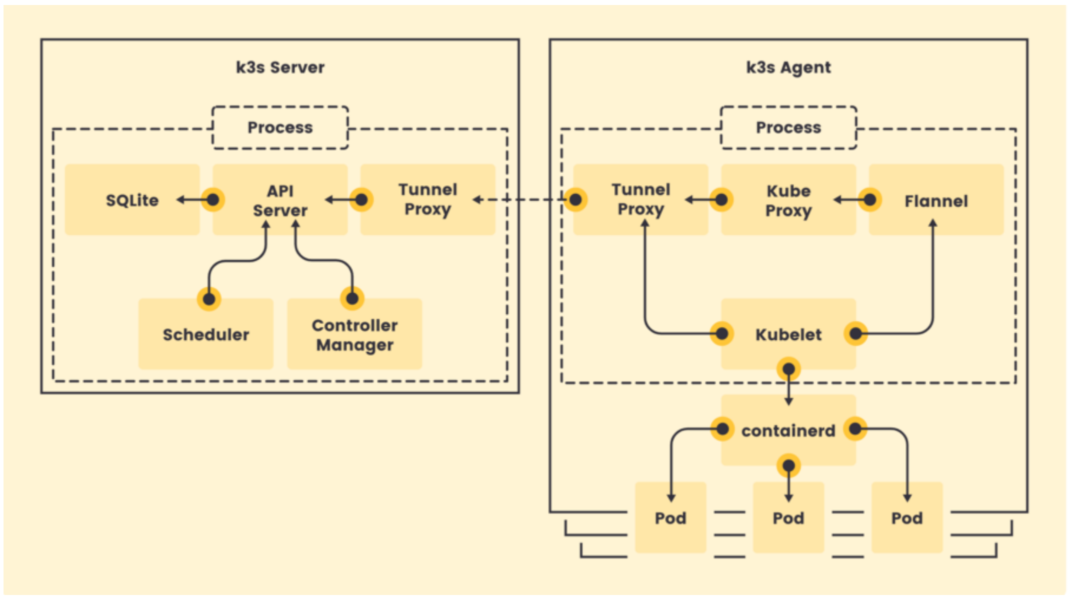


Figure 25: K3s Architecture

1.2.3 OKD

OKD is a cloud-based Kubernetes platform released by Red Hat. OKD is provided as a certified distribution of Kubernetes optimized for continuous application development and multi-tenant deployment. More specifically these optimizations allow users not only to deploy hybrid cloud applications in a cluster but also to distribute multiple clusters in a variety of different public or private cloud platforms and data centres.

This Kubernetes distribution is extended by Red Hat technology and inherits a set of components from Fedora OS which along with Red Hat certifications provide a high-quality assurance perspective although being an open-source software product.

One key difference with other Kubernetes platforms is that OKD uses Fedora CoreOS (FCOS) container-oriented minimal operating system specifically designed to be running containerized applications from OKD.

CoreOS provides multiple capabilities including:

- Ignition: firstboot system configuration for initial deployment.
- CRI-O: a Kubernetes native container runtime implementation closely integrated with the operating system working towards more optimized and efficient Kubernetes experience.
- Kubelet: as explained in section xx, the primary node agent in charge of launching and monitoring containers.

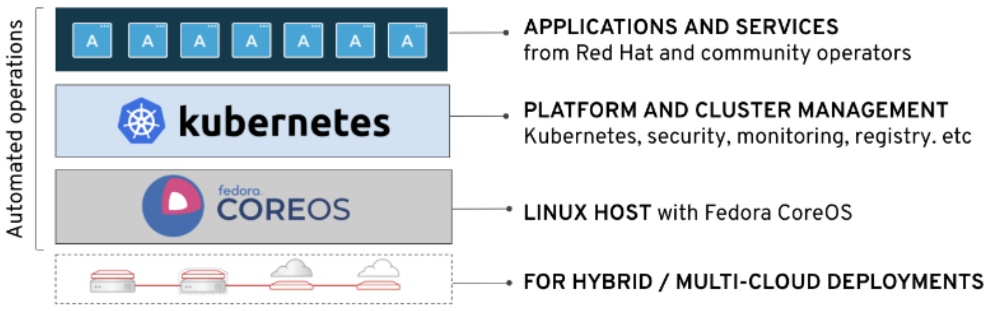

Automated operations
A A A A A A A
kubernetes
fedora COREOS
APPLICATIONS AND SERVICES from Red Hat and community operators
PLATFORM AND CLUSTER MANAGEMENT Kubernetes, security, monitoring, registry. etc
LINUX HOST with Fedora CoreOS
FOR HYBRID / MULTI-CLOUD DEPLOYMENTS

Figure 26: OKD High-level Layers Overview

As part of the enhancements made by Red Hat was the addition of security features and DevOps tools. These security features are provided to ensure the container security and to enable compliance with the most used security and encryption protocols (proxy, CA, nodes, OLM, etc…). As for the DevOps tools, these are oriented to development of applications and operations such as deployment, scaling, and general long-term lifecycle management of these applications. Formally OKD is the open-source sibling of well-known Red Hat OpenShift commercial Kubernetes platform. Red Hat OpenShift is reviewed in the next section.

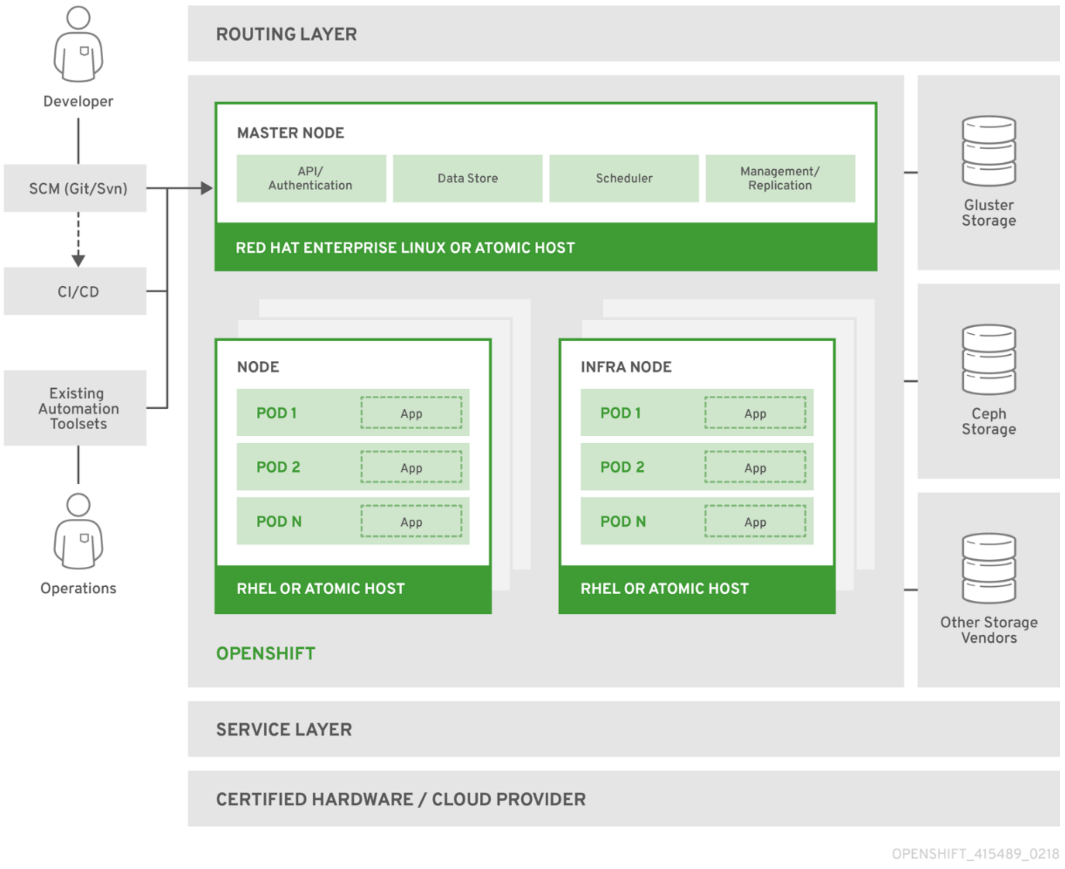


Figure 27: OKD and Openshift Architecture Overview

The basic OKD lifecycle operations could be the following:

- Create OKD cluster.
- Manage the cluster.
- Development and deployment of applications.
- Apply Scaling operations on the cluster.

1.2.4 OpenShift

As illustrated in the Figure 11, OpenShift is a multi-layer platform designed to expose docker based and Kubernetes based solutions. OpenShift adds flexibility and eases continuous development, integrations, and deployments for developers.

OpenShift is composed by the following layers:

- Source code management builds and deployments for developers.
- Managing container images as they are promoted across the system.
- Application management at scale.
- Team and user tracking for a large development structure organization.
- Networking infrastructure support t for clusters.

OpenShift is a microservices based container platform, decoupled in small units that work together independently. This platform runs on top of Kubernetes cluster, using ETCD as main storage system. Services exposed by mentioned cluster are the following:

- REST APIs, exposing core services and objects.
- Controllers, read the APIs that apply changes to objects, communicate and manage these objects.

Objects: high level core concepts that compose OpenShift and provide development-oriented features, including lifecycle management features, e.g., containers, images, pods, services, users, templates, deployments…

OpenShift Platform users consumes the REST APIs to change the state of the system. Controllers use REST APIs to read, and process requested changes from the user and try to accomplish those changes and put the system into synch as expected. In other words, when controllers of different objects detect that changes are requested, they will run processes and perform actions to accomplish a desired state of the system. An example is the build operation, once user have requested build operation, the corresponding build controller will trigger internal actions since a build object was created by the REST APIs. When mentioned actions are done, the controllers will update the system via REST API so the user can see that new build. It should be noticed that this REST API – Controllers pattern makes possible to customize existing features and generate new ones. New controllers can be created with a completely customized "business logic" to perform, this means that any feature with any logical implementation can be provided as ad-hoc. Please, consider that these customizations usually bring synchronization issues and can generate failures capable of turning the system into inconsistent state. To solve this kind of issues, controllers can perform actions like rollback, restart, and resynchronization.

Note: Please, notice that OKD and OpenShift are essentially the same product in terms of architecture, the differences between them reside on service level agreement and support.

Licencing: ODK is licenced under Apache Licence 2.0, OpenShift provides licence for multiple editions with different pricings.

Maintenance: ODK is maintained by its community only, OpenShift is also maintained by dedicated engineering team from Red Hat.

Service Level Agreement: Red Hat provides support team of engineers for OpenShift enterprise users, OKD is only supported by its community through GitHub Issues and mailing list.

1.2.5 Rancher

Rancher [67] is an open-source Kubernetes management tool developed by Rancher Labs in 2018. This tool works on top of Kubernetes clusters providing extra layers for cluster orchestration and federation. Other enhancements are also added, such as improved monitoring system, role-based access, and high compatibility with different providers.

The Figure 28: shows the high-level architecture composed by two fundamental elements:

- Rancher Server: contains all necessary sub-components to manage Kubernetes. These components are explained and highlighted in Figure 7 of this document where Kubernetes Architecture is explained. On top of Kubernetes components Rancher provides authentication proxy which manages role-base access, Cluster controllers that manage underlying clusters and API server to orchestrate all these components.
- User Clusters: these are deployed by user and can be self-provided or provided by services such as Amazon Kubernetes Service. These clusters contain common Kubernetes sub-components (cluster agent, API, kubelet etc…).

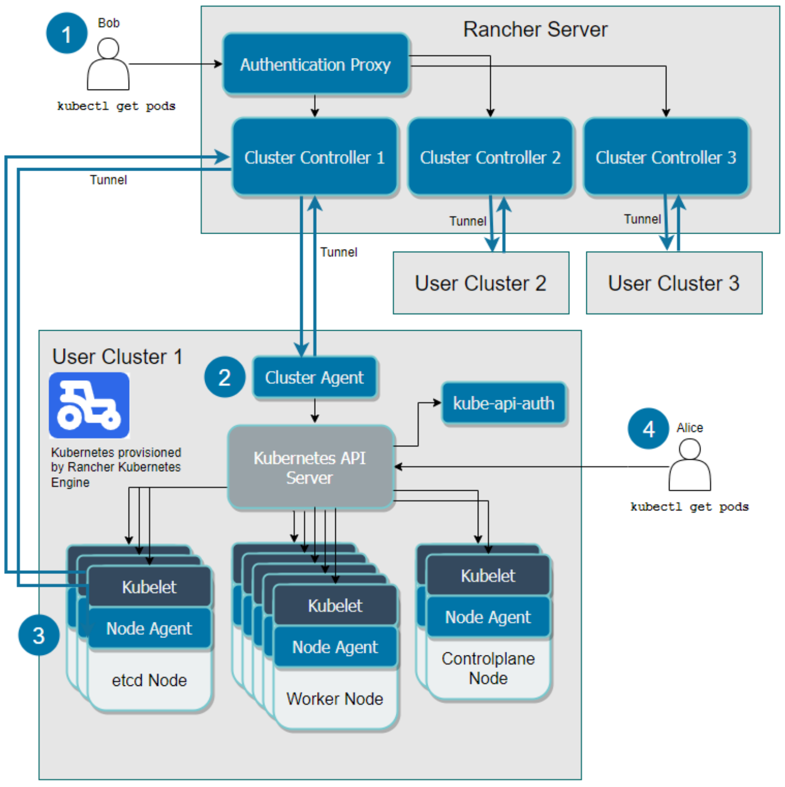


Figure 28: Rancher High-level Architecture

1.2.6 Knative

Knative [68] is and open-source solution release in 2021 with a purpose to be platform-agnostic tool and delivered to build and deploy serverless and event-driven applications. To understand how Knative serves mentioned deployments the following Figure 13 is presented:

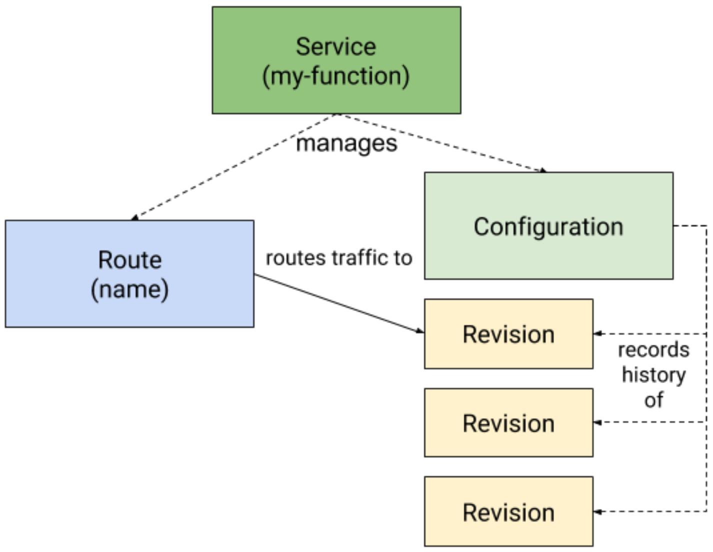


Figure 29: Knative Serving

Knative Serving occurs within a set of Kubernetes Custom Resource Definition (CRDs) objects. These objects will define the workflows and ensure a controlled execution on the cluster.

Knative requires some resources presented as following:

- Services: containing routes and configurations which together provide utilities such as access control, lifecycle management. In a broader view it acts as an orchestration such as Kubernetes ReplicaSet.
- Routes: provide access points such as HTTP endpoint that provides stable and configurable traffic provision. In other words, it is an API in charge of providing the rollout of the latest revision.
- Configurations: creates, describes, and maintain the state of the container images as well as the corresponding metadata so the latest revision state is fulfilled.
- Revisions: as previously mentioned represents a container image with a specific tag or version. In other words, it is a captured picture of the current state code and the corresponding configurations.

Finally, the orchestration is accomplished when changes are applied to the configuration, this triggers the generation of a new revision. This revision acts as a single element that the route will deploy. This revision can be monitored, including historical states, it also provides fundamental operations in long term performance such as rollback and updates.

1.2.7 Open Cluster Management

Open Cluster Management (OCM) is an upstream project that handles multi-cluster (hybrid and multi-cloud) environments for Kubernetes. The project covers cluster registration, work distribution and dynamic placement of policies and workloads.

The project considers two kinds of clusters:

- Hub cluster: Refers to the cluster that runs the multi-cluster control plane of OCM.
- Klusterlets / Managed clusters / Spoke clusters: Refers to the clusters that are managed by the hub cluster. The klusterlet pulls and reconciles the latest configurations from the hub cluster.

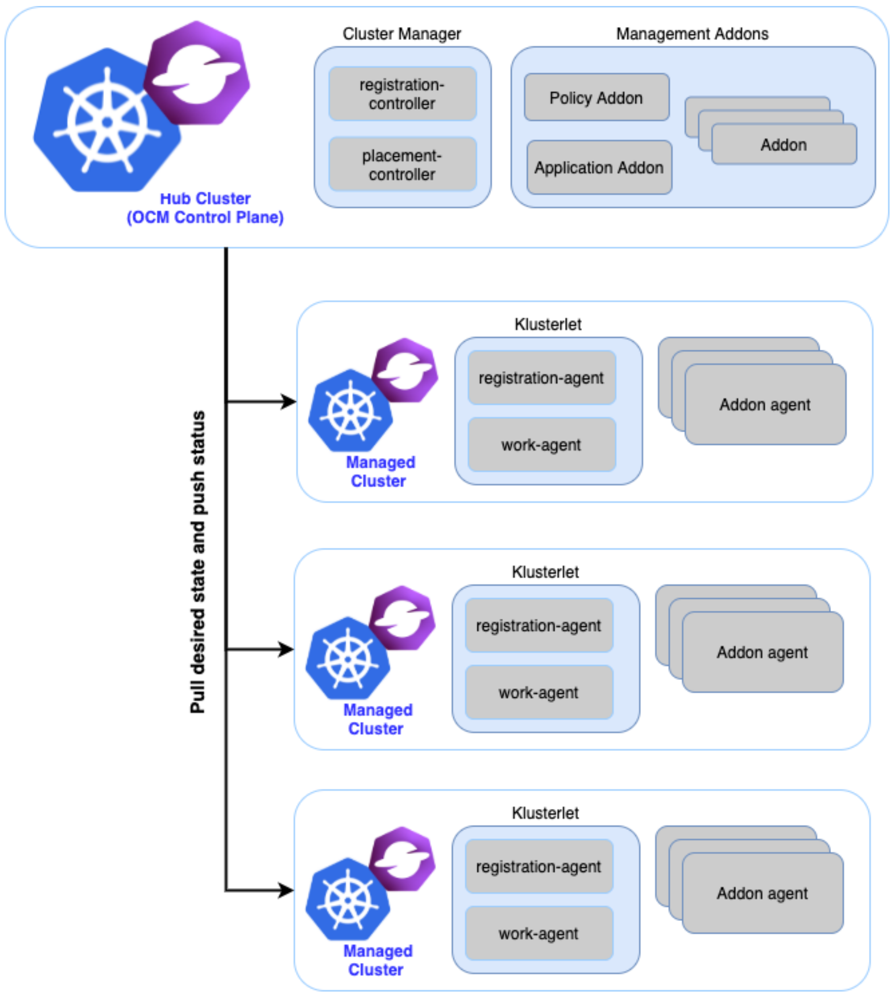


Figure 30: OCM architecture

Some of the main features of the project include:

- Cluster inventory: The ability to register and centrally manage several clusters.
- Work distribution: The possibility to deploy resources into the managed clusters from the hub.
- Content placement: Across the different managed clusters.
- Vendor neutral APIs: Work seamlessly across on premises bare-metal environments and different cloud providers.

2 Networking and Secure Communications

2.1 Submariner

Submariner [2] is built upon several main components:

- Gateway Engine: manages the (secure) tunnels to other clusters.
- Route Agent: routes cross-cluster traffic from nodes to the active Gateway Engine.
- Broker: allows the exchange of metadata between various Gateway Engines enabling them to discover one another.
- Service Discovery: provides DNS discovery of services across the cluster continuum.
- Globalnet Controller: handles interconnection of clusters with overlapping CIDRs.

The diagram below illustrates the basic architecture of Submariner:

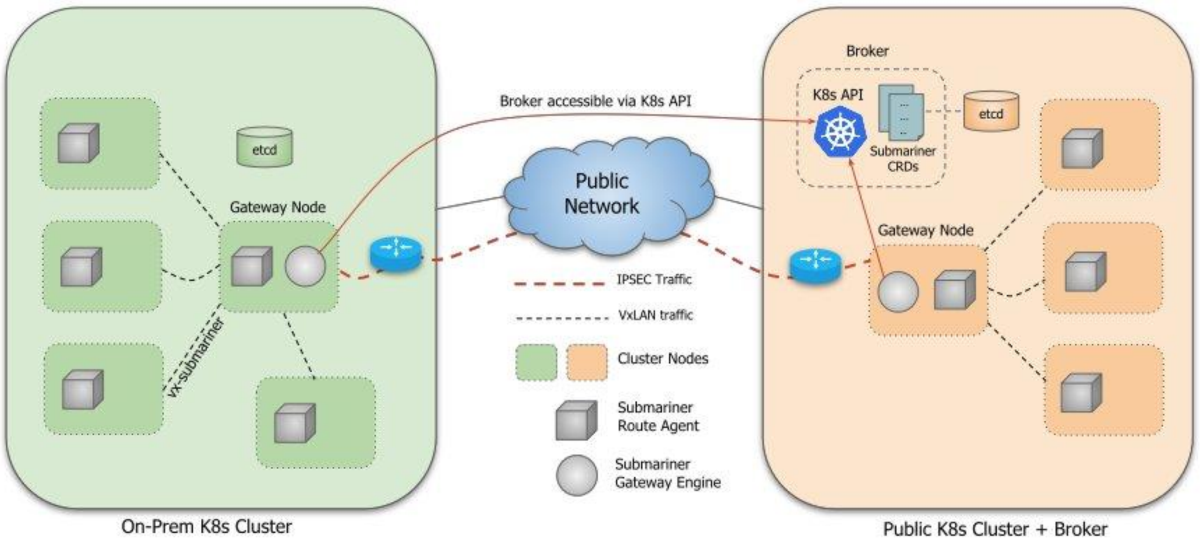


Figure 31: Submariner architecture

Reference: Submariner Architecture

2.2 Mesh Networking (and routing)

A mesh network comprises a type of local area network (LAN) topology, where multiple devices or nodes are connected in a non-hierarchical manner, so that they can cooperate, and provide significant network coverage to a wider area compared to the area coverage achieved by a single router. As mesh networks consist of multiple nodes, responsible for signal sending and information relaying, every node of the network needs to be connected to another via a dedicated link. Since mesh networks leverage on a multi-hop wireless backbone formed by stationary routers, they can connect both mobile and stationary users. Mesh networks have significant advantages such as fast and easy network extension, self-configuration and self-healing and reliability as a single node failure does not result in total network failure. Several types of metrics have been proposed for mesh networks, as summarized below:

- Number of hops related metrics (Hop Count metrics)
- Connection quality related metrics (Link Quality Metrics)
- Network load related metrics (Load-dependent metrics)
- Multi-channel metrics

In general, routing protocols for mesh networks can be divided in two main categories: (i) proactive and (ii) reactive. Proactive protocols expect all network nodes to continuously maintain at least one routing table, where information related to the routes to each of the other network nodes is stored and recurrently broadcast them along the network to exchange and update information of the routing tables stored in neighbouring nodes. On the other hand, reactive protocols follow an on-demand strategy, in the sense that information exchange is taking place only in cases of a packet transmission. Although reactive protocols are associated with less additional network traffic generation, they are also associated with more time needed to forward data, since before the actual data exchange information about the available route must first be exchanged. Some well-established reactive protocols include the Ad hoc On-Demand Distance Vector Routing Protocol (AODV) - which supports unicast, multicast and broadcast communications -, the Optimized Link State Routing (OLSR) protocol - which uses a hop-by-hop strategy and the Dynamic Source Routing (DSR) protocol – which is ideal for multi-hop wireless ad-hoc networks.

On the other hand, well-established proactive routing protocols include the Better Approach to Mobile Ad-hoc Networks – advanced (BATMAN-adv) - which is based on enabling each node to determine its best next hop, without knowledge of the entire network topology requirement, thus leading to a significant reduction of control messages in the network - and Babel – which is based on the Bellman-Ford algorithm.

Regarding network topologies supporting Internet of Things (IoT) devices, they are in general either star or tree-based, by enabling data collection by sensor groups and then transmitted to a centralized entity, where centralized processing occurs. Since mesh topologies enable the dynamic connection of network nodes in a non-hierarchical and dynamic way, this topology constitutes a perfect candidate for IoT networks.

3 Far Edge Device Orchestration and Management

3.1 Genie

Jobs

Job is a main concept of Genie [4]. Genie provides 2 ways to execute a job:

- Submit via API (delegating execution to the server)
- Execute locally via agent CLI

API jobs are easier to submit with any REST client. For example, the user can just say "Run query Q with SparkSQL". The server takes care of all the details, the user can sit back, wait for the job to complete and then retrieve the output via API. The diagram below presents the lifecycle of an API job.

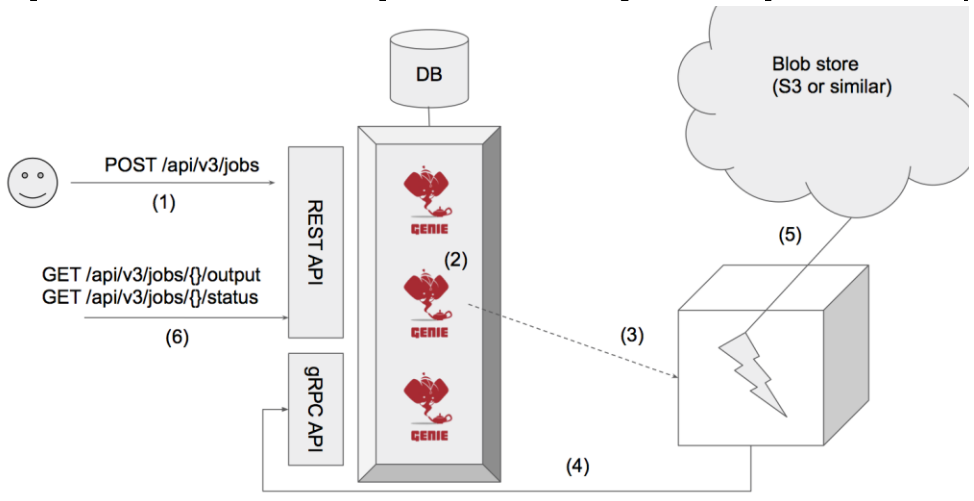


Figure 32: Lifecycle of an API job

Genie can be viewed as a federated Big Data orchestration and execution engine.

3.2 Nuvla

Once installed, NuvlaEdge[6] is a turn-key solution. From factory settings, you plug it in, power it up and you are good to go. The automated and secured registration process ensures that each edge device is yours and uniquely configured and initialised. This can even include an on-demand remote secured VPN access gives you access to your devices and applications as you need it.

The Nuvla platform exposes a powerful REST API. This API allows developers to integrate Nuvla into third-party systems, script it and even use it as Infrastructure as code (IaC).

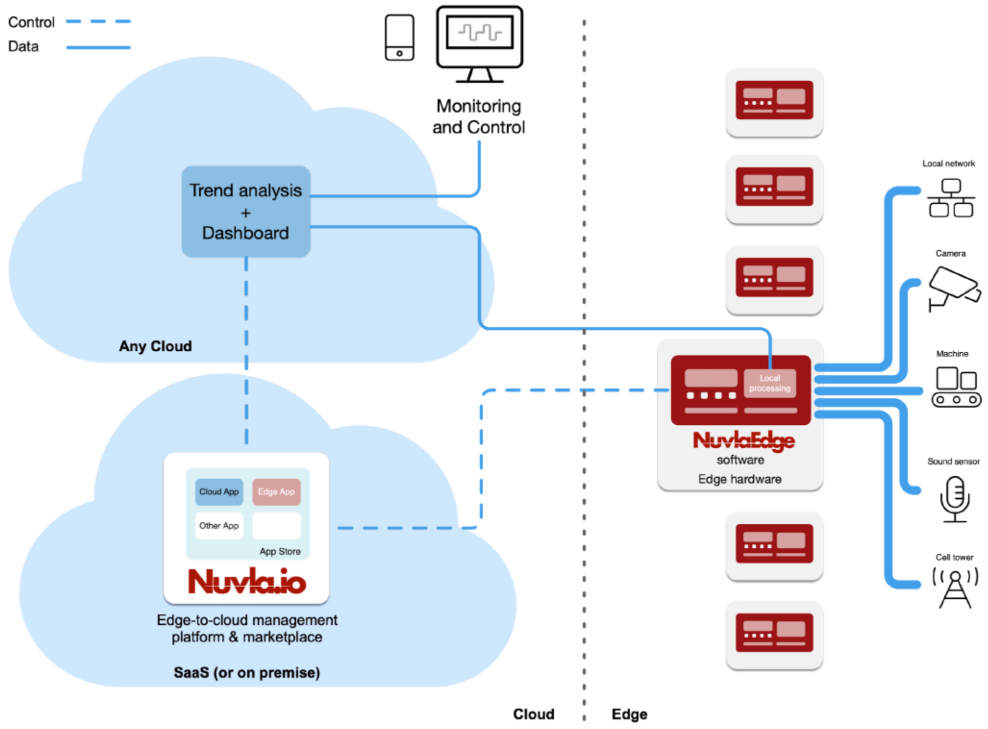


Figure 33: Nuvla diagram

3.3 HPE GreenLake [7]

This highly agile "cloud everywhere" modality allows for the integration of on-premises and/or private cloud deployment models into a centralized HPE GreenLake interface for managing those resources. Within GreenLake, users can access a myriad of deployable resources and services: bare metal, compute, storage, container management and data protection services, as well as HPC, AI/ML and virtual desktop infrastructure, just to name a few.

HPE GreenLake customers can select from a broad range of pre-configured, fully customizable end-to-end modular solutions that offer choices of infrastructure components (e.g., storage, compute, networking, containers, ML, private cloud and more) customized to their specific needs, expertise level and amount of available in-house IT resources--in fact, HPE GreenLake supports almost any HPE technology. Users simply select a preconfigured offering and size, order with a few clicks, and await delivery to their data center, colocation facility or edge location. Upon arrival, installation is plug-and-play with a quick, simple configuration process.

3.4 openEuler

System Framework

openEuler is used mainly for servers. It consists of the basic acceleration library, virtualization, kernel, driver, compiler, system tool, OpenJDK, and other components.

As the one OS for all scenarios, openEuler has an innovative architecture and full-stack optimizations to unleash the full computing power of diversified architectures.

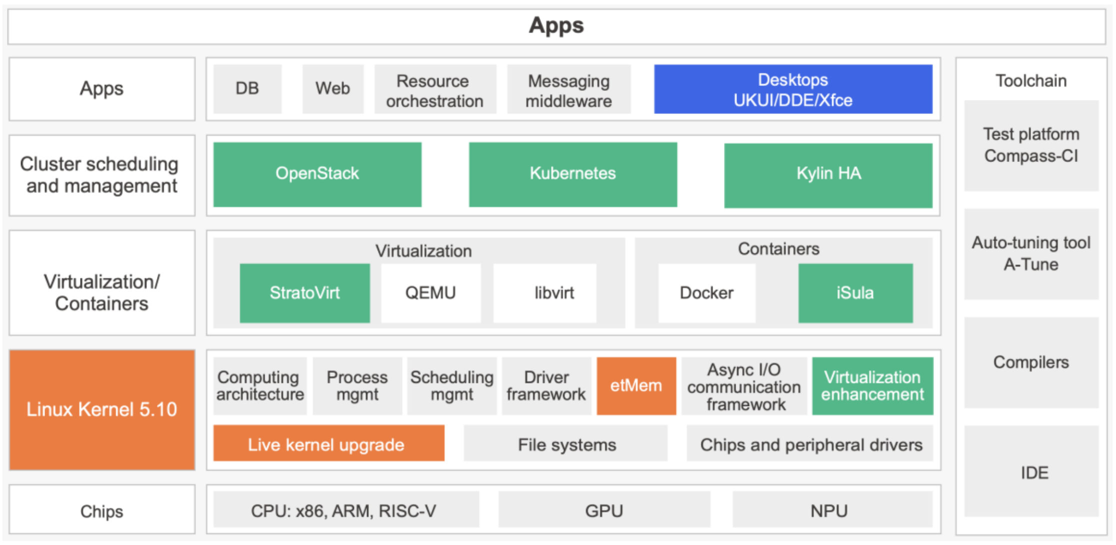


Figure 34: openEuler architecture

Since November 2021, the openEuler platform is developed and operated by OpenAtom Foundation (a non-profit organization which promotes free software in China).

3.5 AWS IoT GreenGrass

The AWS IoT Greengrass [8] client software, also called AWS IoT Greengrass Core software, runs on Windows and Linux-based distributions, such as Ubuntu or Raspberry Pi OS, for devices with ARM or x86 architectures. With AWS IoT Greengrass, users can program devices to act locally on the data they generate, run predictions based on machine learning models, and filter and aggregate device data. AWS IoT Greengrass enables local execution of AWS Lambda functions, Docker containers, native OS processes, or custom runtimes of user's choice.

AWS IoT Greengrass provides pre-built software modules called components that let user easily extend edge device functionality. AWS IoT Greengrass components enable to connect to AWS services and third-party applications at the edge. After user develops IoT applications, AWS IoT Greengrass enables user to remotely deploy, configure, and manage those applications on the fleet of devices in the field.

The following example shows how an AWS IoT Greengrass device interacts with the AWS IoT Greengrass cloud service and other AWS services in the AWS Cloud.

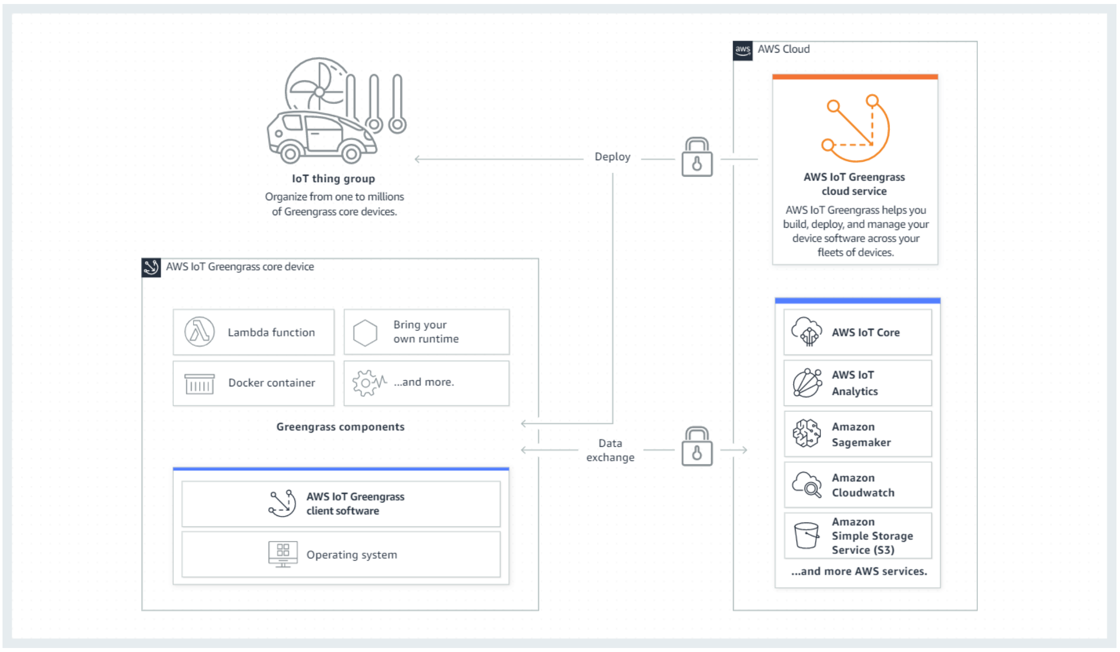

AWS Cloud
IoT thing group
Organize from one to millions of Greengrass core devices.
Deploy
AWS IoT Greengrass cloud service
AWS IoT Greengrass helps you build, deploy, and manage your device software across your fleets of devices.
AWS IoT Greengrass core device
Lambda function
Bring your own runtime
Docker container
...and more.
Greengrass components
Data exchange
AWS IoT Greengrass client software
Operating system
AWS IoT Core
AWS IoT Analytics
Amazon Sagemaker
Amazon Cloudwatch
Amazon Simple Storage Service (S3)
...and more AWS services.

Figure 35: Interaction between AWS IoT Greengrass and the AWS Cloud

4 Function Management – FaaS

4.1 Openwhisk [9]

The OpenWhisk platform supports a programming model in which developers write functional logic (called Actions), in any supported programming language, that can be dynamically scheduled and run-in response to associated events (via Triggers) from external sources (Feeds) or from HTTP requests. The project includes a REST API-based CLI along with other tooling to support packaging, catalogue services and many popular container deployment options.

Apache OpenWhisk supports many deployment options both locally and within Cloud infrastructures. Options include popular Container frameworks such as Kubernetes and OpenShift, and Compose. The community endorses deployment on Kubernetes using Helm charts since it provides easy and convenient implementations for both Devlopers and Operators.

The main entities of the platform are described below.

Actions

Actions are stateless functions (code snippets) that run on the OpenWhisk platform. Actions encapsulate application logic to be executed in response to events. Actions can be invoked manually by the OpenWhisk REST API, OpenWhisk CLI, simple and user-created APIs or automated via Triggers.

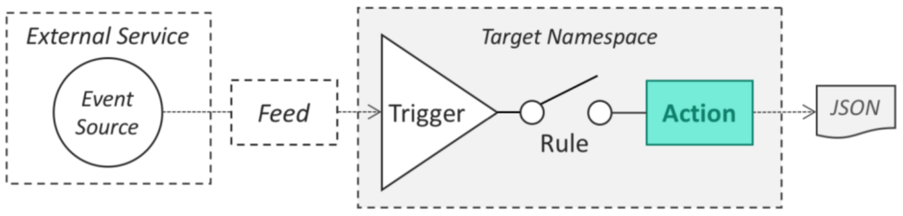


Figure 36: Openwhisk Action Definition.

An action may be created from a function programmed using several supported languages and runtimes,

- Go
- Java
- JavaScript
- PHP
- Python
- Ruby
- Rust
- Swift
- .NET Core
- Docker and native binaries

or from a binary-compatible executable, or even executables packaged as Docker containers.

Triggers and Rules

OpenWhisk triggers and rules bring event-driven capabilities to the platform. Events from external and internal event sources are channelled through a trigger, and rules allow your actions to react to these events.

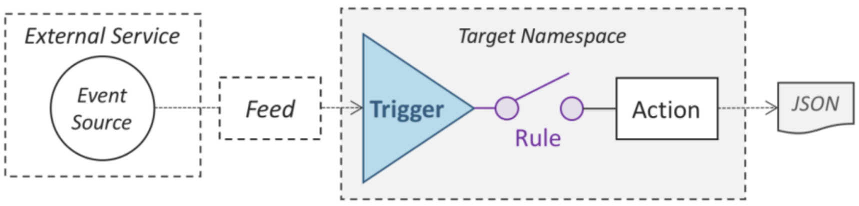


Figure 37: Openwhisk Trigger Definition.

Triggers are named channels for classes or kinds of events sent from Event Sources. Rules are used to associate one trigger with one action. After this kind of association is created, each time a trigger event is fired, the action is invoked.

Feeds

OpenWhisk supports an open API, where any user can expose an event producer service as a feed in a package. Feeds are intended for advanced OpenWhisk users who intend to publish their own feeds. There are three architectural patterns for creating a feed: Hooks, Polling and Connections.

Hooks

In the Hooks pattern, we set up a feed using a webhook facility exposed by another service. In this strategy, we configure a webhook on an external service to POST directly to a URL to fire a trigger. This is by far the easiest and most attractive option for implementing low-frequency feeds.

Polling

In the "Polling" pattern, we arrange for an OpenWhisk action to poll an endpoint periodically to fetch new data. This pattern is relatively easy to build, but the frequency of events will of course be limited by the polling interval.

Connections

In the "Connections" pattern, we stand up a separate service somewhere that maintains a persistent connection to a feed source. The connection-based implementation might interact with a service endpoint via long polling, or to set up a push notification.

4.2 OpenFaaS [10]

OpenFaaS makes it easy for developers to deploy event-driven functions and microservices to Kubernetes without repetitive, boiler-plate coding. Below a number of highlights is provided:

- Ease of use through UI portal and one-click install.
- Allows to
  - write services and functions in any language with Template Store or a Dockerfile,
  - build and ship user’s code in an OCI-compatible/Docker image.
- Portable: runs on existing hardware or public/private cloud by leveraging Kubernetes.
- CLI available with YAML format for templating and defining functions.
- Auto-scales as demand increases including to zero.
- Active community.
- Community Edition for developers.
- Pro edition & support for production.
- Commercially supported distribution by the team behind OpenFaaS.

Conceptual workflow

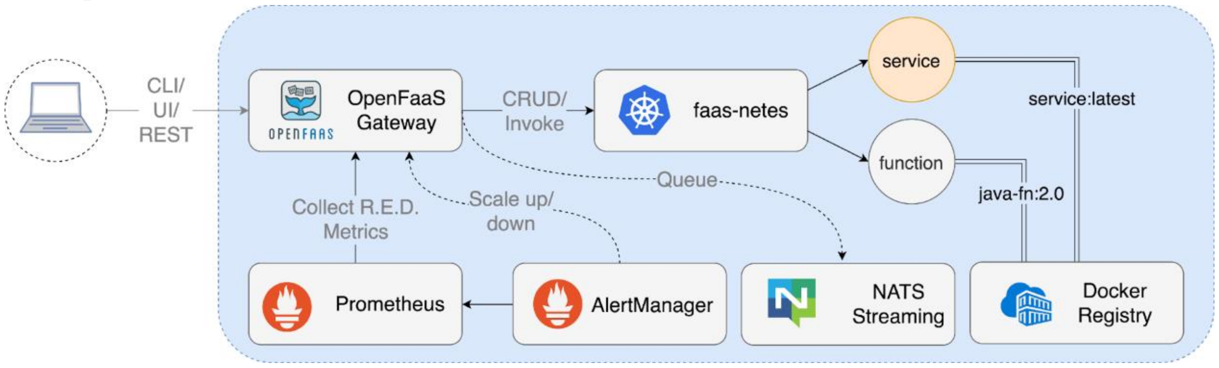


Figure 38: Openfaas workflow

The Gateway can be accessed through its REST API, via the CLI or through the UI. All services or functions get a default route exposed, but custom domains can also be used for each endpoint. Prometheus collects metrics which are available via the Gateway's API and which are used for auto-scaling. By changing the URL for a function from /function/NAME to /async-function/NAME an invocation can be run in a queue using NATS Streaming. One can also pass an optional callback URL.

faas-netes is the most popular orchestration provider for OpenFaaS, but Docker Swarm, Hashicorp Nomad, AWS Fargate/ECS, and AWS Lambda have also been developed by the community. Providers are built with the faas-provider SDK.

5 AI-assisted continuum management

5.1 Federated Learning Benchmarking: Baselines on the continuum and open-source tools

Federated learning is valuable for training machine learning models on decentralized data sources. It offers several advantages over traditional centralized training methods, including improved privacy, scalability, efficiency, and robustness. Figure 1 depicts the case where the data Owners train their local data within each local cluster A, B and C onto models A, B and C, which are ultimately shared into the aggregation method to update the global model on the architecture.

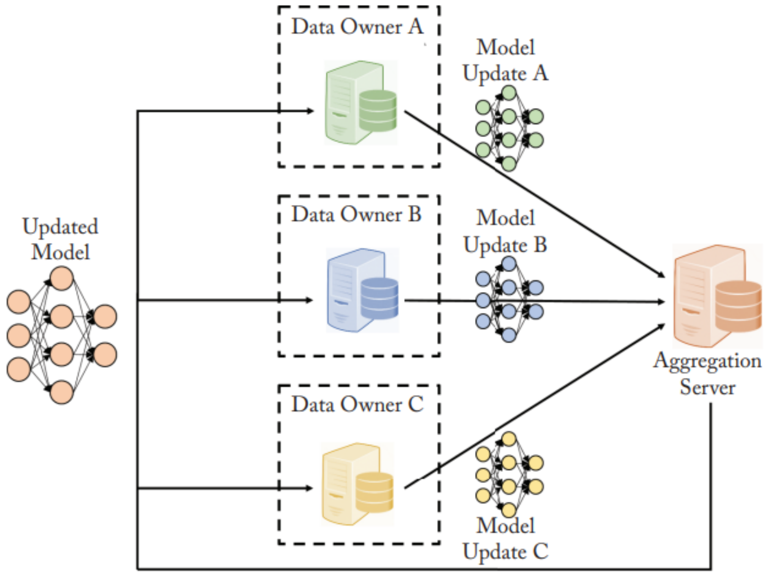


Figure 39: A Client-Server federated learning architecture[70]

As introduced by [71] on a federated learning benchmark, three categories for FL frameworks are identified:

- **All-in-one framework**: concentrates on diverse use-cases and is frequently built with significant engineering effort (e.g., maintained by industrial institutions), these include: FATE, FedML [72], PaddleFL, Fedleaner.
- **Horizontal-only framework**: These frameworks attempt to provide simple-to-use APIs for users to adopt and build horizontal FL algorithms rather than supporting various applications with horizontal FL. Enabling instance, TFF from Google offers federated learning API and federated core API for users to apply and build FL algorithms, respectively. These APIs are built on TensorFlow: TFF, Flower, FLUTE[73].
- **Specialized frameworks**: Designed for specific purposes.
  - CrypTen focuses on providing secure multi-party computation primitives [74]
  - FedTree is designed for the federated training of decision trees [75].

From the previous, we have identified the following as open-source FL frameworks that could operate in the continuum [76]:

- **Fate(Federated AI Technology Enabler)**: It is an open-source framework supported by WeBank that is accessible in standalone and cluster setups but necessitates personal modification and understanding of protocol buffers. FATE currently offers a variety of federated learning algorithms, including logistic regression, tree-based algorithms, deep learning, and transfer learning, to support diverse federated learning situations.

- **Baidu PaddleFL**: Based on Paddle's large-scale distributed training and elastic scheduling of training jobs on Kubernetes, PaddleFL can be easily deployed based on full-stack open-sourced software and applications such as traditional machine learning training strategies such as Multi-task learning, Transfer Learning in Federated Learning settings.
- **Nvidia Clara SDK**: It includes full-stack GPU-accelerated libraries, SDKs, and reference applications for developers, data scientists, and researchers to create real-time, secure, and scalable federated learning solutions.
- **Flower**: The design of Flower is designed to be customizable because Federated learning systems vary wildly from one use case to another. Flower can be used with any machine learning framework, for example, PyTorch, TensorFlow, Hugging Face Transformers, PyTorch Lightning, MXNet, scikit-learn, JAX, TFLite, or even raw NumPy for users who enjoy computing gradients by hand. It allows customizing the strategy, initializing parameters on the server side, choosing a different strategy, and evaluating models on the server side
- **FedML**: The federated learning and analytics library enables secure and collaborative machine learning on decentralized data anywhere at any scale, supporting large-scale cross-silo federated learning, cross-device federated learning on smartphones/IoTs, and research simulation.
- **Pysift**: PySyft is an open-source Python 3 based library that enables federated learning for research purposes and uses FL, differential privacy, and encrypted computations using PyTorch.
- **Tensorflow Federated (TFF)**: TensorFlow Federated (TFF) is a Python 3 open-source framework for federated learning developed by Google.
- **IBM Federated Learning**[free-trial]: Its goal is to give federated learning a strong foundation that supports a wide range of federated learning models, topologies, learning models, etc., especially in business and hybrid-Cloud environments. Concerns about privacy and secrecy, the need for regulatory compliance, and the usefulness of shifting data to a single, central location for learning are the key forces underpinning FL.

Main Federated Learning aggregation baseline methods to apply in the continuum:

- Communication-Efficient Learning of Deep Networks from Decentralised Data (**FedAvg**): This method is the Federated Learning baseline that consists of a set of clients that update a model locally and then transmits it to the server, while the server aggregates the model weights from local clients and then sends the aggregated model back to them with respect to their numbers of training samples and transmits the aggregated one back to the clients[77].
- Federated Optimization in Heterogeneous Networks (**FedProx**): This method (Li et al., 2020) is the FL baseline, which proposes a regularisation term that minimizes the weight differences between local and global models, which prevents the model from diverging from the local training data. However, when the local data is extremely heterogeneous, it is more appropriate to train a personalized model for each client collaboratively rather than learning a single global model [78]
- Federated Learning with Personalization Layers (**FedPer**): This method (Arivazhagan et al., 2019) is a personalized method that can deal with extremely heterogeneous networks, where it is proposed to train a single model per client which shares only the base layers while having local, personalized layers for each client, to keep the local knowledge. So it does not share the personalized layers [79].
- Federated Graph Neural Network for Privacy-Preserving Recommendation (**FedGNN**): This method is the subgraph FL baseline, which proposes to share nodes between the local client and the nodes in other clients by expanding the local subgraph using augmented nodes from other clients that satisfy similar node features [80].
- Subgraph Federated Learning with Missing Neighbor Generation (**FedSage+**): This approach is a subgraph FL baseline that also aims to jointly train a potent and generalizable graph mining model without directly transferring their graph data. They suggest two solutions: (1) FedSage, which

develops a GraphSage [81] model based on FedAvg to integrate node features, link structures, and task labels on numerous local subgraphs; and (2) FedSage+, which develops a missing neighbour generator alongside FedSage to address missing links between local subgraphs [82]. The graph generator estimates the gradient of distances between the transmitted node representations and the other clients' node characteristics before transmitting the local node representations to other clients in order to train.

- **FedOpt**: Adaptive Federated Optimization (AFO) is a federated learning algorithm that aims to improve the convergence and performance of federated learning by adaptively selecting the best-performing devices for each iteration of training. This is done by using a selection function that evaluates the quality of the local models on each device and selects the best-performing ones for training the global model. This can help to improve the quality of the global model by ensuring that it is trained on the most accurate local models. In addition to adaptively selecting the devices for training, AFO also uses momentum terms and a learning rate schedule to improve the convergence of the algorithm. The momentum terms help to accelerate the training of the global model by incorporating information from previous iterations, while the learning rate schedule allows the learning rate to be adjusted based on the performance of the model. Overall, AFO is a promising approach for improving the performance of federated learning by adaptively selecting the best-performing devices and using advanced optimization techniques. However, more research is needed to understand the effectiveness of this algorithm fully and to compare it with other federated learning approaches. By using a simple client/server optimizer framework, FedOpt incorporates adaptivity into FL in a principled, intuitive, and theoretically-justified manner [83]
- Split Federated Learning (**SplitFL**): This algorithm is similar to FedAvg, but instead of using a global model that is updated by each device, SplitFL uses two models: a global model and a local model. The global model is trained on the global dataset, while the local model is trained on the local dataset on each device. The local models are then averaged to produce a new global model, which is used to update the local models on each device. Moreover, the split model makes SL a better option for resource-constrained environments. However, SL performs slower than FL due to the relay-based training across multiple clients [84]
- Federated Transfer Learning (**FedTL**): This algorithm is used when more than the data on each device is needed to train a good model. In this case, a pre-trained global model is used as a starting point, and the local models on each device are fine-tuned on their local datasets. The local updates are then averaged to produce a new global model, which is used to update the local models on each device. The FedTL employs the Paillier homomorphic encryption approach to safeguard the data privacy of the social computing nodes. To cut down on the expenses associated with computing and communication in the federated learning system, a unique application of transfer learning technology is implemented. [85]
- Federated Dual Learning (**FedDL**): This algorithm is used when the data on each device is labelled in a different way, such as in the case of different languages. In this case, two models are trained on the federated dataset: one model is trained to predict the labels on one device, and the other model is trained to predict the labels on the other device. The two models are then combined to produce a new global model, which is used to update the local models on each device. Specifically, they create a dynamic layer-sharing method that combines models in an iterative, bottom-up way as it learns the similarity between users' model weights to construct the sharing structure. [86]
- Federated Evolutionary Learning (**FedEL**): This algorithm uses evolutionary algorithms, such as genetic algorithms, to train models on the federated dataset. In this approach, a population of models is initialized and then trained on the local datasets on each device. The local updates are then combined using evolutionary algorithms to produce a new global model, which is used to update the local models on each device. [87]

5.2 Transfer Learning

Transfer learning is increasingly employed as a method during the development process of a machine learning model when the domains it was trained on are similar and can be used to develop similar tasks [88]. This method of developing machine learning models requires fewer resources and less labelled data to train new models, and most importantly, it avoids the task of having to train a neural network from scratch. The ongoing improvement of these models will heavily rely on transfer learning. Examples include enhancing network effectiveness, optimizing marketing campaigns for a higher return on investment, and promoting the advancement of speech recognition software. In a variety of industries, complex tasks are completed using machine learning algorithms. The Figure 40 shows the learning case when using traditional and transfer learning approaches, (a) depicts the general case for algorithms that learn everything from scratch, while in (b), we can observe the transfer-learning approach, where the knowledge is transferred from a source that already has the knowledge for those tasks.

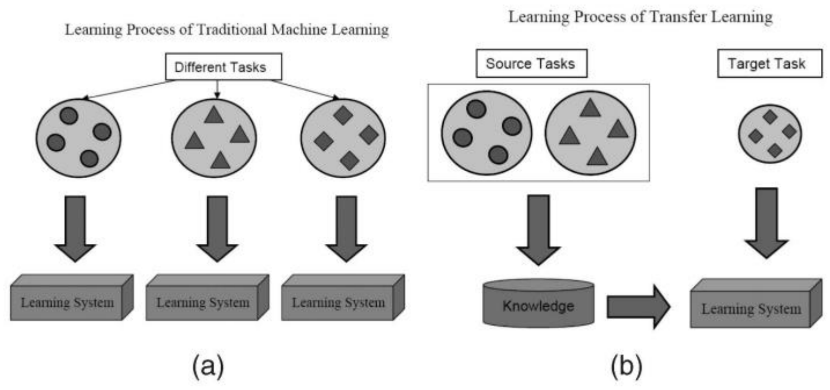


Figure 40: Learning case when using traditional and transfer learning approaches

There exist multiple libraries for doing transfer-learning on a Machine Learning task. We have identified the following open-source options:

- **Hugging-Face Hub**: Described before in the AI marketplace as also containing models that can be downloaded for performing transfer learning tasks.
- **Torchvision.models**[20]: To address various tasks, such as image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection, video classification, and optical flow, Pytorch has its own implementation of pre-trained models that are currently implemented on torchvision. This module also contains torch.hub[21], which currently provides APIs to explore all available models using torch.hub.list().
- **Tensorflow datasets**[22]: It contains a big variety of models stored in the following locations:
  - **The TensorFlow hub**, a repository of pre-trained models for TensorFlow ready for fine-tuning and deployment.
  - **Model Garden**, a GitHub repository highly specialized in TensorFlow's high-level APU.
  - **TensorFlow.js models**, a collection of Tensorflow models that can run on any web browser.
- data. Despite being open-source, it contains a limited number of neural network architectures that can be used for transfer learning.

[20] https://pytorch.org/vision/stable/models.html

[21] https://pytorch.org/docs/stable/hub.html#module-torch.hub

[22] https://www.tensorflow.org/resources/models-datasets

- **Segmentation_models.pytorch**[23]: This is an open-source library containing several pre-trained segmentations models in Pytorch that integrates more than 400 encoders from Timm[24].
- **Paperswithcode**[25]: The platform Papers-with-code enables searching through a large collection of articles that have been published along with their code implementation. Furthermore, it includes a number of cutting-edge ML models, including as well pre-trained ML models.
- **Modelzoo**[26]: Model Zoo allows filtering models by keywords, tasks, and frameworks using a nice, simple interface. For Tensorflow, PyTorch, Caffe, and other programs. Additionally, the license and requirements for the majority of the models are on GitHub.

5.3 Model Optimization Techniques

These techniques are used to improve the performance and efficiency of machine learning models. These techniques can be applied to trained models to reduce their size, improve their accuracy, or make them more efficient for running on different devices.

Pruning and distillation are two popular model optimization techniques commonly used in the industry. Pruning involves removing unnecessary connections and parameters from a trained model, which can reduce the size of the model and make it more efficient to run. Distillation involves training a smaller model to mimic the behaviour of a larger, pre-trained model, which can also reduce the size of the model and improve its accuracy.

These techniques can be particularly useful for training and deploying machine learning models on constrained devices, such as smartphones or IoT devices. These devices often have limited computational resources and storage, which can make it challenging to run large and complex models. By applying pruning and distillation techniques, the model can be made smaller and more efficient, which can enable it to run on these devices.

Pruning and distillation are valuable model optimization techniques commonly used in the industry. They can be applied to train and compress models for operation in constrained devices, which can improve their performance and efficiency.

5.3.1 Pruning Techniques

By utilizing effective sparse connectivity, good performance may be preserved even when certain connections are eliminated, which might be crucial for modeling high-performance models on edge nodes. Pruning is a technique for minimizing over-parameterized networks while maintaining high accuracy. Normally, it employs mask arrays to reduce magnitude weights during training or inference. Pruning is a model optimization technique that involves removing unnecessary connections and parameters from a trained model. This can reduce the size of the model and make it more efficient to run.

Pruning can be applied to a variety of different model architectures, including feedforward neural networks, convolutional neural networks, and recurrent neural networks. It is often used in conjunction with other model optimization techniques, such as quantization or distillation, further to improve the performance and efficiency of the model. Up to 90% of trained networks' parameters can be reduced by using pruning techniques [90]. This shows how pruning facilitates the discovery of winning tickets (critical weights) for expedited generalization.

[23] https://github.com/qubvel/segmentation_models.pytorch

[24] https://rwightman.github.io/

[25] https://paperswithcode.com/

[26] https://modelzoo.co/

There are several different approaches to pruning a model, including weight pruning, unit pruning, and structural pruning. Weight pruning involves removing individual weights or connections from the model that have a small magnitude, while unit pruning involves removing entire units or neurons from the model. Structural pruning involves removing entire layers or blocks of the model; other techniques include

- **Local pruning:** It consists of an iterative pruning on a neural network, where each layer is processed individually in terms of weight magnitude, activation, gradient, etc.
- **Global pruning:** It consists of pruning the entire model at once by removing a percentage of the slowest connections.
- **Custom pruning:** This method permits to customize of the pruning technique by altering, e.g. the way the masks are being applied during the pruning procedures. An example is provided by Pytorch Subclassing with BasePruningMethod[27].

Example of pruning tools:

Prune torch module: Pytorch exemplifies different pruning techniques[28] based on LeNet architecture from LeCun et al., 1998 [91]**.** Customization is open-source but requires manual inspection of each desired architecture.

Ml-kit[29]: In contrast to rival offerings, ML Kit is a mobile SDK that supports Android and IOS apps for on-device processes offered powered by Google's best-in-class ML models and offered at no cost[30] for commercial and general purposes. The data is locally rather than sending it to a server for processing and then delivering the findings for the application to utilize. On-device data processing has the following advantages: reduced latency, real-time operation without network lag, offline functionality, and increased data security

5.3.2 Network Distillation

The distillation techniques have been successfully used for application-specific, and resource-constrained IoT platforms Joint training and distillation approaches follow a teacher-student learning strategy, which generally uses a larger teacher network to train a compact student network (on-device model) with minimal loss in accuracy. Together with transfer learning, which deals with the transfer of knowledge learned from one domain to another domain, network distillation holds the substantial potential to significantly reduce model size without compromising performance in terms of accuracy[31]. It has also been shown that transferring the knowledge from an ensemble of nodes or from a large, highly regularized model into a smaller, distilled model can achieve equally high performance, but most importantly, knowledge distillation can reduce the training times of a full ensemble considerably, making implementations much easier to deploy [92]. As can be seen from Figure 41, a teacher model is distilled, and then a student model tries to mimic the teacher model using knowledge distillation.

[27] https://pytorch.org/docs/stable/generated/torch.nn.utils.prune.BasePruningMethod.html

[28] https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

[29] https://developers.google.com/ml-kit/

[30] https://developers.google.com/ml-kit/guides

[31] https://keras.io/examples/vision/knowledge_distillation/

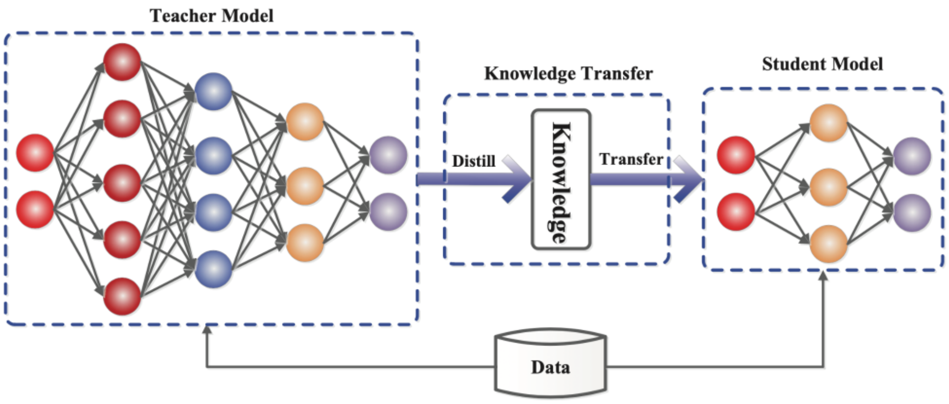


Figure 41: The teacher-student framework for knowledge distillation[93]

The distillation techniques can be divided, as shown in Figure 42 that is, into three categories, namely, response, feature, and relation-based knowledge.

- **Response-based knowledge:** In this type of learning, the student model attempts to learn from the predictions of the teacher model, and to do so, a distillation loss can be defined to minimize the distance between the logits of both distillation methods [94]
- **Feature-based knowledge:** This method tries to minimize the feature differences between the teacher and student model during training by using the intermediate layers so that the student model tries to learn the same neuron activations as the teacher model [95]
- **Relation-based knowledge:** This method is based on calculating the similarity between feature maps, similarity matrixes, or embeddings so that the relationship between the features between the teacher and student model can also be learned at different representations [96]

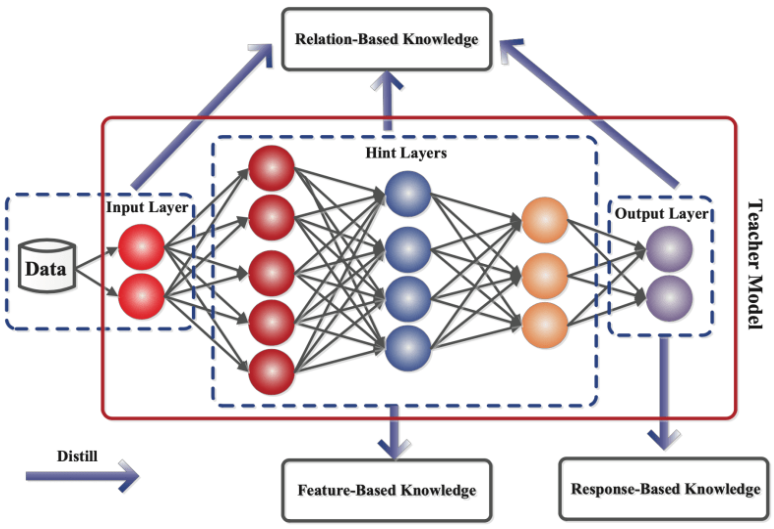


Figure 42: The different kinds of knowledge in a teacher model[93]

Algorithms for knowledge distillation:

- **Adversarial distillation:** The generative adversarial logic is used to teach the student network to discriminate generated data from the real one, so that it can learn a better representation of the data [97]
- **Multi-teacher distillation:** Knowledge is obtained from multiple teacher models, similar to an ensembling model mainly based on response and feature-based knowledge by using the intermediate layers of each model.
- **Cross-modal distillation:** The knowledge of a teacher model trained on one modality is distilled to teach a student model the knowledge of another modality. This method can be useful for models such as image-text captioning.

Other categories of algorithms include Graph-based distillation, Attention-based distillation, Data-free distillation, Quantized distillation, Lifelong distillation and Neural architecture search-based distillation.

Some examples of a distillation model are **DistilGPT2** (short for Distilled-GPT2), which is an English-language model that was pre-trained under the direction of the tiniest iteration of Generative Pre-trained Transformer 2. (GPT-2)[32]. **Neural Network Distiller**[33] by Intel AI Lab: a Python package for neural network compression research. **Text brewer**[34]: A PyTorch-based knowledge distillation toolkit for natural language processing.

5.3.3 Hyperparameter Tuning

Besides GridSearchCV[35] and RandomSearch[36], the following are open-source frameworks that can also be used for hyper-tuning and auto-tuning:

- **Ray**[37]: A unified framework for scaling Python and AI applications is called Ray. For the purpose of streamlining ML computation, Ray consists of a core distributed runtime and a toolkit of libraries (Ray AIR).
- **AutoKeras**[38] is an open-source library for Automated Machine Learning (AutoML).
- **Optuna**[39]: Optuna is a software framework for automated hyperparameter optimization that was created specifically for machine learning. It has a user API that is imperative and define-by-run. The Optuna user can dynamically create the search spaces for the hyperparameters, and Optuna code benefits from high modularity thanks to our define-by-run API.
- **HyperOpt**[40]: Python's Hyperopt library allows for serial and parallel optimization over challenging search spaces, including those with conditional, discrete, and real-valued dimensions.
- **Scikit Optimize**[41]: It is a Python hyperparameter optimization open-source package. The Scikit-learn development team created it. Compared to other hyperparameter optimization packages, it is rather simple to use.

[32] https://huggingface.co/distilgpt2?text=My+name+is+Lewis+and+I+like+to

[33] https://intellabs.github.io/distiller

[34] https://github.com/airaria/TextBrewer

[35] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[36] ttps://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[37] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[38] http://autokeras.com/

[39] https://github.com/optuna/optuna/blob/dafd0ccdc85f5b2d185264a4b9debf3643db64fe/docs/source/index.rst

[40] https://github.com/hyperopt/hyperopt

[41] https://scikit-optimize.github.io/stable/auto_examples/hyperparameter-optimization.html

- **Microsoft's NNI (Neural Network Intelligence)**[42]: Microsoft created NNI, a free and open-source AutoML toolkit. It is employed to automate hyper-parameter tuning, model compression, and search for neural architectures.

5.3.4 Clustering methods for resource management

The classification is two-fold: identifying the algorithms that facilitate the fog/edge computing, and secondly, the architectures and infrastructures that facilitate the resource management on the network as it is shown in Figure 43.

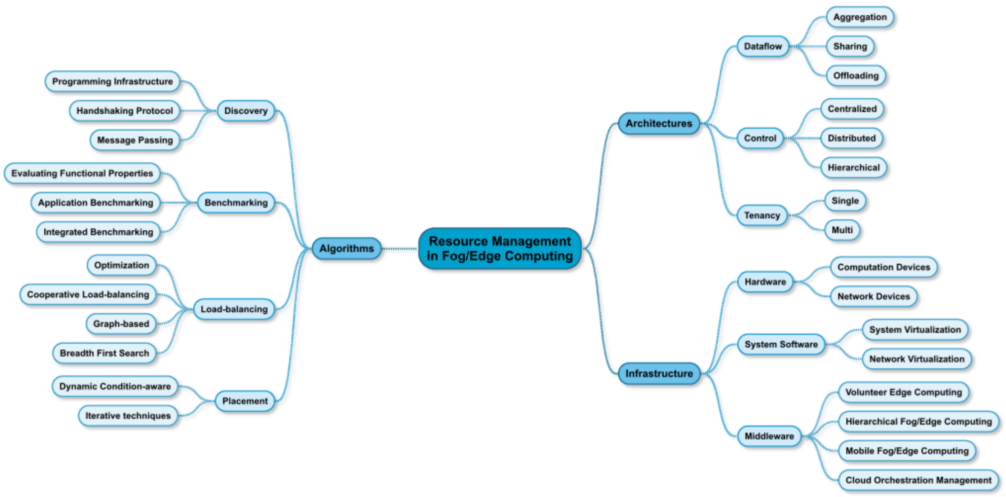


Figure 43: Architectures & algorithms for resource management on a fog/edge computing model [98]

1. **Algorithms that facilitate resource management on the network**: This includes the following categorization (see Figure 44):
   a. **Discovery**, where the goal is to identify the available edge resources on the network making use of message passing, handshaking protocols, and programming infrastructure; This is based on the direction of the data movement between the cloud and the edge nodes within the ecosystem, encompassed by aggregation, sharing, and offloading techniques.;
   b. **Load-balancing**, which refers to the algorithms that would help to distribute the tasks appropriately across the edge network, methods that include Optimization techniques, cooperative load-balancing, graph-based methods & bread First Search;
   c. **Benchmarking**, The performance of an algorithm can be analysed using performance metrics, which are quantitative measures of its efficiency and effectiveness. These metrics include throughput, latency, utilization, and others. Power consumption and memory usage can also be considered when analysing performance. By combining these metrics, a comprehensive view of the performance can be obtained, providing insight into areas for improvement and optimizing resource usage in a system. Performance metrics are important tools for evaluating algorithms.
   d. **Placement**: Placement of computation tasks on suitable fog/edge resources in a dynamic and changing environment is a challenging issue in fog/edge computing. Placement algorithms are used to address this issue by considering the availability of resources and environmental factors such as device location, network characteristics, and current workload. These algorithms must

[42] https://www.microsoft.com/en-us/research/project/neural-network-intelligence/

be adaptive to changes in these factors to maintain system performance. Effective placement algorithms are crucial for ensuring computation tasks are placed on suitable fog/edge resources.

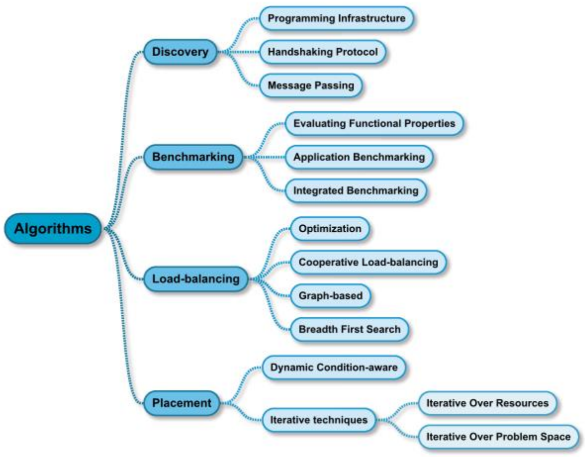


Figure 44: Algorithms classification for a fog/edge computing model [98]

2. **Architectures and infrastructure in the Fog/Edge commuting: This is two-fold, including the infrastructure**, which includes the hardware and middleware, and secondly, techniques based on resource management, which has been identified in the ground of the data flow, control and tenancy architectures.

Architectures for resource management.

1. **Dataflow architectures**: Based on the direction of the data movement between the cloud and the edge nodes within the ecosystem, encompassed by aggregation, sharing and offloading techniques.
    a. **Aggregation**: This type of model usually allows an edge model to collect data generated from multiple tenancies and then tries to reduce unnecessary information, including extra metadata (traffic, buffers, etc.). Amongst the main techniques, the following are the most commonly used: Graph-based techniques, Cluster-based techniques, Petri Net-based techniques, Decoupled Techniques, Batch Techniques, Batch techniques and Hybrid techniques
    b. **Sharing**: This technique works under the umbrella of Mobile Cloud Computing (MCC) [99], where the workload is shared among peers on a network, meaning that at any moment, any device on a mobile network may leave silently at any time. The classification can be explained as follows:
        i. **Based on control**: Includes centralized and distributed control. On the first one, a centralized controller is in charge of managing the workloads of each edge device on the network, while on the second, the goal is to provide each edge device with the capability to control its own workload independently.
        ii. **Based on adaptive Techniques**: These techniques include multiple objectives while employing a shared model, where the goal is to optimize each workload depending on the consumption of resources: energy and minimal latencies, for example. We can categorize it into four types, depending on the connectivity between a device and its neighbours and using shortcuts such as calculating the shortest path amongst them: i) Connectivity aware, ii) Heterogeneity-aware, iii) Security–aware, and iv) Fairness-aware
        iii. **Based on Cooperation**: Using Ad-hop Cooperation, the nodes can be used as a cooperation technique to share workloads on a device-to-device communication. Alternatively, using infrastructure-based cooperation might be more effective for a more tightly coupled environment.

c. **Offloading**: This method moves the application or data into the edge device and can happen in two different ways: 1) from the user device to the edge, where brute force, greedy heuristic, simulated annealing, and fuzzy logic are the most common approaches as it can be seen from Figure 45. 2) from the Cloud to the Edge, including techniques such as including server offloading, caching mechanisms, and web programming.

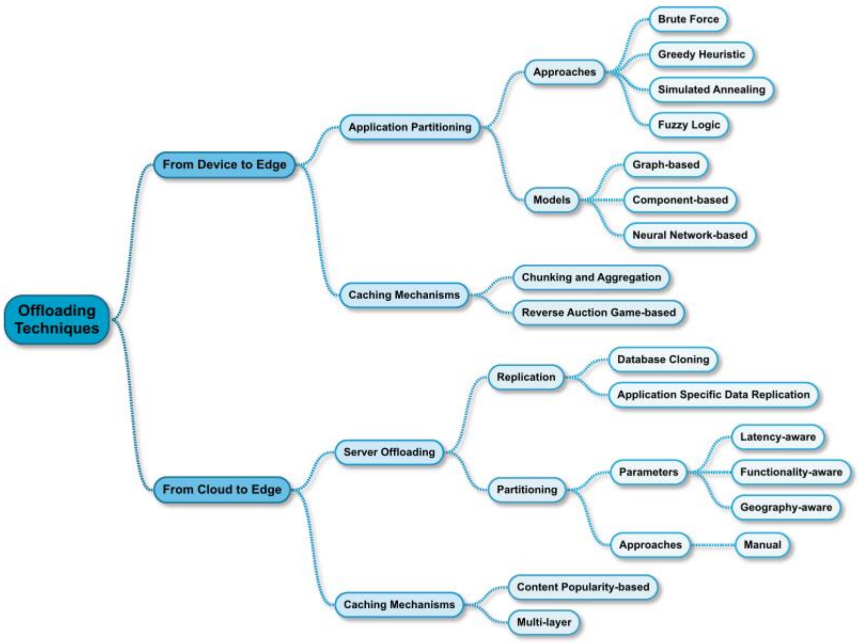


Figure 45: Offloading techniques for resource management [98]

2. **Control architectures**: These architectures are based on the way the resources are controlled within the ecosystem, e.g. through a controller. In the context of fog/edge computing, control architectures refer to the overall design and structure of the system that is used to manage the allocation and use of resources such as computing power, storage, and network bandwidth. These architectures typically involve the use of various algorithms and protocols to distribute tasks and workloads among the different components of the system in order to optimize performance and ensure efficient resource utilization.

   One common classification of control architectures for resource management in fog/edge computing is based on the level of decentralization and autonomy of the system. For example, some architectures may be highly centralized, with a single central controller that makes all decisions about resource allocation and management. Other architectures may be more decentralized, with multiple controllers working together to manage resources in a distributed manner.

   Another way to classify control architectures for resource management in fog/edge computing is based on the specific algorithms and techniques used to manage resources. For instance, some architectures may use traditional optimization algorithms, such as linear programming or dynamic programming, to determine the best allocation of resources. Others may use more sophisticated techniques, such as machine learning or artificial intelligence, to adapt to changing workloads and optimize resource utilization in real time.

   a. **Centralized**: These types of control architectures usually are defined as (1) Solver-based, which uses an optimization objective that tries to minimize the costs and optimize the network costs, and (2) Graph Matching-based, which focuses on a minimum weight-matching problem.

   b. **Distributed**: These methods include

      i. **Blockchain-based** methods for distributed peer-to-peer networks without intermediaries.

ii. **Game Theoretic Approach-based**, This approach is a type of control architecture that uses game theory to solve the problem of task offloading in a multi-channel wireless interference environment [100]. The game theoretic approach is useful because it allows for distributed decision-making, which can be more efficient and scalable than centralized approaches

iii. **Genetic Algorithm-based**, which uses a computing node to perform a task-map graph that minimizes resource consumption.

iv. **Sensor Function Virtualization-based**: SFV is a concept in which decision-making can be modularized and deployed on multiple nodes in an IoT network. This allows for greater flexibility and adaptability, as modules can be added at runtime on different nodes. However, SFV is still in the early stages of development and has not yet been demonstrated in a real-world IoT testbed.

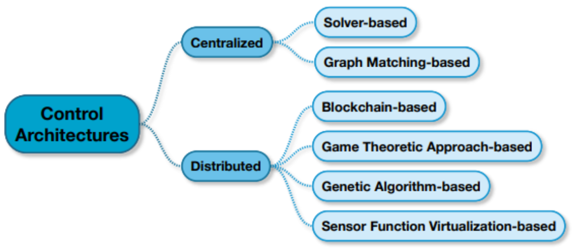


Figure 46: Classification of control architectures [98]

The choice of a control architecture for resource management in fog/edge computing can have a significant impact on the performance and efficiency of the system. Therefore, it is important for designers and developers to carefully consider the trade-offs and potential benefits of different architectures in order to select the one that is most suitable for their specific application and use case.

3. **Tenancy architectures**: For this case, we only have single and multiple-tenancy. These architectures are designed to support multiple tenants that are consuming the same resource for the case of multi-tenancy. On the single-tenancy, single-tenant hardware is consumed by a unique entity.

Infrastructure for resource management.

Infrastructure categorization: The infrastructure can be categorized into hardware, System Software and Middleware.

- The hardware is of great importance to exploit the small computational resources that small-form-factor-devices can provide, including network gateways, Wifi Access Points, small servers, set-top boxes, cars and so on [101].
- System software, which refers to the system software to be deployed on the edge devices, such as containers and virtual machines, encompassed by the system virtualization. It also includes network visualization, which contains Software-defined Networking (SDN) & Network Function Virtualization (NFV), as well as overlay networks. The systems software defines the operating system requirements, as well as the virtualization software.
- The Middleware will coordinate distributed computing nodes and would deploy them to the fog/edge cloud. It also focuses on complementary services that are not supported by the system software

6 Data Processing Tools

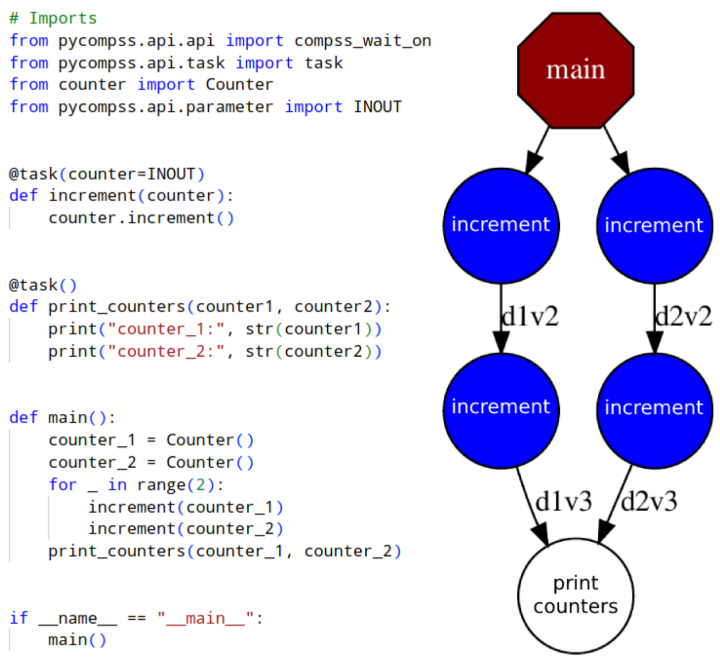
In the continuum, data processing is needed generally in three different scenarios that are approached with a particular technological solution. In the first scenario, called sense-process-actuate, one of the devices of the infrastructure senses an event and the IT platform is expected to process it and provide an appropriate response even activating some of the actuators to produce some physical effect. Generally, services falling in this scenario turn to Function-as-a-Service solutions to support their operation. Whereas in the first scenario data processing is triggered by an eventual data generation, the second scenario considers the processing of a continuous generation of data (stream). This continuous processing aims at updating the data stored in the system or producing other streams of data with the results. Dataflow Managers are frameworks that ease the development of services exploiting this second scenario by connecting the processing of the different streams. In contrast with the aforementioned scenarios where the processing is triggered by data, the third scenario considers data-at-rest; computations are triggered by the system administrator or the end user to run compute-heavy processes to analyse large amounts of data or run large simulations. This is the traditional case of batch jobs in distributed computing handled by general-purpose workflow managers and domain-specific distributed programming frameworks. This diversity in programming frameworks and models hinders the efficient development of solutions targeting the continuum. The following subsections cast a glance over different frameworks supporting the mentioned technologies.

6.1 Batch Processing

6.1.1 COMPSs

COMPSs is a programming framework that aims at easing the development and execution of distributed applications. The core of the framework is its task-based programming model which allows developers to program their applications as a sequential code targeting a single compute node with a widely-adopted general-purpose language like Python[], C++ or Java. At execution time, a runtime engine decomposes the application into many parts (tasks), analyses the dependencies among them to find the inherent parallelism, and exploits it by distributing their executions over the underlying infrastructure such as Clusters, Supercomputers or in the Cloud-to-thing continuum dealing with the necessary data transfers.

Tasks are the unit of parallelism and distribution and they correspond to the execution of certain functions of the code selected by the programmer through annotations. Every time that the code of the application reaches a call to an annotated method, its execution is transparently replaced by a request to the runtime system to run the method asynchronously on any node of the infrastructure. For detecting potential data dependencies with other asynchronous tasks, the runtime must be aware of what data values are involved in the task – i.e., the arguments of the specific call, the callee object and the return values – and how the method operates on them – creation, reading, or update– which needs to be specified by the programmer. The task parameters can be objects in memory, files, or a special data type (collections) which enable the detection of data dependencies considering individual elements within a set of values. The programming model syntax is completed with a small set of API calls, mainly focusing on synchronisation. The leftmost part of Figure 47 illustrates an example of a COMPSs application written in Python.



```python
# Imports
from pycompss.api.api import compss_wait_on
from pycompss.api.task import task
from counter import Counter
from pycompss.api.parameter import INOUT


@task(counter=INOUT)
def increment(counter):
    counter.increment()


@task()
def print_counters(counter1, counter2):
    print("counter_1:", str(counter1))
    print("counter_2:", str(counter2))


def main():
    counter_1 = Counter()
    counter_2 = Counter()
    for _ in range(2):
        increment(counter_1)
        increment(counter_2)
    print_counters(counter_1, counter_2)


if __name__ == "__main__":
    main()
```

Figure 47: Example of a COMPSs application code and the corresponding task graph

Besides the code of the invoked method, tasks can have alternative implementations. Such implementations can be other methods of the application, methods programmed in other languages, binary executions, mpi runs or Docker or Singularity containers.

Upon the analysis of the accesses, the runtime system adds the task to a directed graph describing the workflow of the application. The nodes of such a graph represent tasks and the directed edges depict a data dependence among the involved tasks. Using this graph, the runtime system is able to identify which tasks are dependency-free and, hence, can start executing over the infrastructure. The graph illustrated in the rightmost part of Figure X corresponds to the previous example code. As soon as the runtime system adds a task to the graph, it starts considering offloading the task to another node of the infrastructure.

To achieve high performance, the runtime allocates resources exclusively to run each task. To that end, it has an internal subcomponent, the Task Scheduler, that monitors the availability of the resources of the underlying infrastructure and submits the execution of the tasks in parallel always ensuring the sequential consistency of the main code (guaranteeing the fulfilment of the data dependencies) and trying to optimize multiple aspects of the execution such as the overall execution time, the energy consumption or the carbon footprint. The runtime system will handle all the necessary data transfers automatically to ensure that all data is available on the assigned host by moving data on demand between the computing nodes.

The COMPSs environment also offers two higher-level APIs to develop applications: DDS, a library for High-Performance Data Analytics, and disLib, a library providing several mathematical and machine learning algorithms with an interface compatible with scikit-learn

6.1.2 Dask

Dask is a python library that enables parallelization using a high-level interface to operate over standard Python collections such as arrays (NumPy), bags (lists) and machine learning datasets(sci-kit learn). In addition, it also offers a low-level interface that enables the parallelization of functions in two modes. With the lazy mode, the programmer indicates with an API call that the calculation of a function is delayed, i.e. it is not computed at the moment and generates; but it generates a future object as result. Delayed operations are executed when the developer programmatically requests the computation of the object through another API call. The real-time mode works in a similar way; however, tasks can start their computation as soon as they are detected.

6.1.3 Spark

Apache Spark is a programming framework for large-scale data analytics. Spark is built on the concept of a distributed dataset: a large collection of arbitrary objects stored in a distributed manner across a large cluster. The building blocks of Spark are called Resilient Distributed Datasets (RDDs). These allow two types of operation on top of the distributed dataset: transformations, which define a new dataset, and actions, which start an execution on the underlying cluster.

The framework provides other higher-level APIs leveraging RDD. For instance, DataFrame organizes the DataSet in named columns and allows relational operations on both external data sources and Spark's built-in distributed collections without providing specific procedures for processing data. Another example is MLlib which provides many distributed ML algorithms covering feature extraction, classification, regression, clustering, recommendation, and more.

6.1.4 Parsl

Parsl is a parallel programming library for Python very similar to COMPSs. Developers annotate Python functions to specify which applications compose their software and indicate opportunities for concurrent execution. These applications can either be Python functions or external binaries. Invocations to these applications become tasks that can be connected by shared input/output data (Python objects or files). Considering these shared values, Parsl constructs a graph of tasks to manage the potential concurrency throughout the execution to enable dynamic workflows determined at execution time while ensuring an efficient execution. Parsl scripts are able to scale from several cores on a single computer to hundreds of cores across a cluster, container orchestrators and clouds.

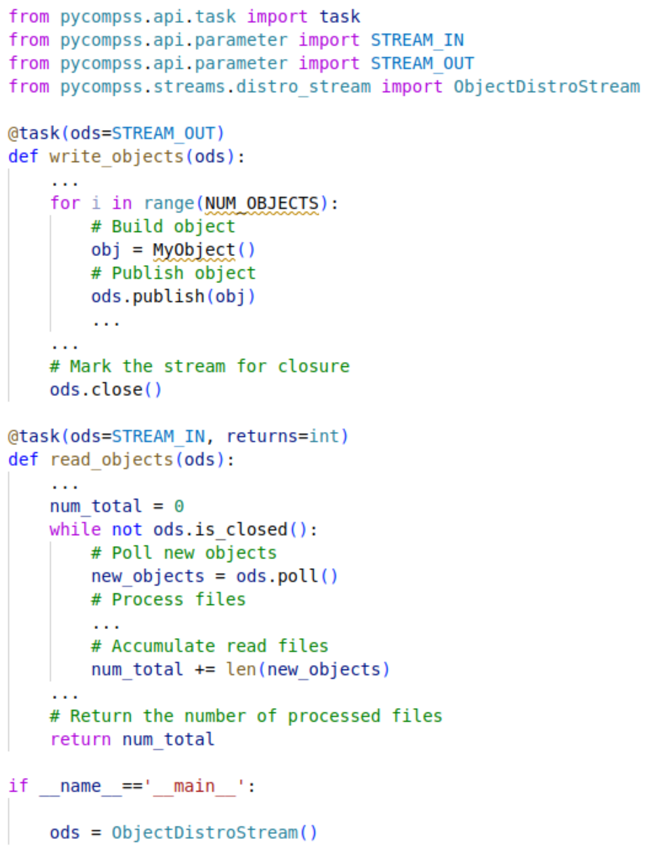
6.1.5 Apache Airflow

Apache Airflow is an open-source dataflow management platform. Similar to PyCOMPSs, Airflow users describe their dataflows programmatically via Python scripts and the runtime generates a DAG to manage the scheduling. Unlike COMPSs, Airflow allows these DAGs to run on a defined schedule (daily or hourly) or triggered by an external event.

6.2 Stream Processing

6.2.1 COMPSs

COMPSs is a programming framework that aims at easing the development and execution of distributed applications already described in the section 6.1 of the Annexes. Besides, the traditional workflow approach with atomic tasks, COMPSs also supports persistent tasks that operate on streams. Thus, applications using the COMPSs programming model can be expressed as a dataflow and combine these persistent tasks with traditional workflows. For that the application developer can create a new ObjectDistroStream or a FileDistroStream and use it in the context of any task specifying the type of the parameter as a STREAM_IN or STREAM_OUT



```
from pycompss.api.task import task
from pycompss.api.parameter import STREAM_IN
from pycompss.api.parameter import STREAM_OUT
from pycompss.streams.distro_stream import ObjectDistroStream

@task(ods=STREAM_OUT)
def write_objects(ods):
    ...
    for i in range(NUM_OBJECTS):
        # Build object
        obj = MyObject()
        # Publish object
        ods.publish(obj)
        ...
    ...
    # Mark the stream for closure
    ods.close()

@task(ods=STREAM_IN, returns=int)
def read_objects(ods):
    ...
    num_total = 0
    while not ods.is_closed():
        # Poll new objects
        new_objects = ods.poll()
        # Process files
        ...
        # Accumulate read files
        num_total += len(new_objects)
    ...
    # Return the number of processed files
    return num_total

if __name__=='__main__':

    ods = ObjectDistroStream()
```

Figure 48: COMPSs example using streams

6.2.2 Apache Beam

Apache Beam is an open source unified programming model to define and execute data processing pipelines. Although the framework was originally targeting dataflows, the framework also supports the execution of batch jobs thanks to its support for multiple runners (distributed processing backends). Its key benefits are the unification of batch and stream processing, the portability of the applications on different runtimes and an API with a high level of abstraction. However, these benefits come at the expense of some performance; directly using the underlying runners would achieve higher-performance.

6.2.3 Eclipse Zenoh-Flow

Eclipse Zenoh-Flow is a data flow programming framework that facilitates as well as structures the development and the deployment of applications across the Cloud-to-Thing continuum.

A Zenoh-Flow application consists of nodes interconnected with links. An application is thus viewed as a directed graph. These graphs are described in a human-readable descriptor file — a contract — that Zenoh-Flow enforces. Starting from this descriptor file, the Zenoh-Flow instantiates the application, placing its components across the infrastructure.

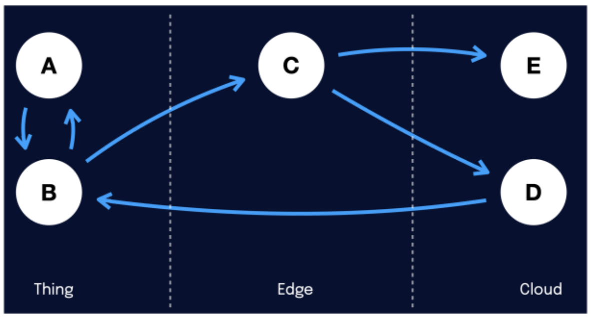


Figure 49: Zenoh-Flow dataflow graph over the Cloud-to-Things continuum

Hence, in Zenoh-Flow, the role of an application developer is limited to: (i) creating the different nodes that compose the application and (ii) describing the connections between them. All other steps are left to the Zenoh-Flow framework.

Zenoh-Flow's main features are:

- A declarative approach. Through its descriptor file, application developers precisely know the structure of what will be deployed and how it will be connected. Its human-readable format has the added advantage of lowering the entry barrier for non-purely technical application designers.
- Optimised communications. Being aware, through the descriptor file, of the application's topology, Zenoh-Flow will adapt the communication channels: nodes located on the same Zenoh-Flow runtime will communicate through specific channels that negligible overhead.
- Unified abstractions. Regardless of where a node will be deployed, a developer only must code it once: Zenoh-Flow nodes implement a unique interface. The result is a shared library that is later dynamically loaded by a Zenoh-Flow runtime.
- Location transparency. As Zenoh-Flow uses Zenoh as its communication medium to cross the runtime boundary, application developers do not need to know ahead of deployment where their nodes will be running. Communications are based on key expressions that Zenoh routes transparently. This also allows us to provide redundancy and load-balancing at low costs.
- Data isolation. Zenoh-Flow associates to each instance of an application a unique identifier that is automatically added and leveraged in the communications. This ensures that if the same application is deployed twice on the same infrastructure or if several applications use the same "topics", no collision will occur. The same technique is used for each link, further allowing nodes to expose the same key expressions.
- High-performance. Benchmarks show that Zenoh-Flow can achieve high throughput and low latency. A port of an Autonomous Driving System over to Zenoh-Flow further illustrated its capabilities, allowing for real-time control of a car in a simulated environment.

To summarise, Zenoh-Flow aims at providing an easier yet more robust and efficient framework to creating distributed applications: thanks to its declarative nature, it can optimise the communications and enforce a known, expected flow of data; thanks its unified abstraction and to its use of Zenoh, developers do not have to worry about where their nodes will be deployed; thanks to its data isolation principle, developers do not have to worry about the topic names their nodes use.

6.2.4 Celery[43]

Celery is an open-source distributed task queue system. Messages are passed through message brokers like RabbitMQ and Redis. Celery supports real time processing as well as scheduling. The Celery system can consist of multiple workers and brokers, enabling high availability and horizontal scaling. Celery comes with a web dashboard and other tools for management and an overview of the system. Celery will be used, among others, as part of the ICOS Security Layer as a log anomaly detection distributed task queueing system, distribution of log anomaly detection training and inference jobs.

6.3 Function-as-a-Service

The section 8.4 Function Management – FaaS already discusses some popular frameworks delivering Function-as-a-Service. This section extends that list with FaaS solutions that natively support the execution of functions programmed following the programming models discussed above and are able to handle the execution of the application across the Continuum.

6.3.1 Colony

Colony is a framework to develop applications running throughout the whole Cloud-Edge Continuum. The framework proposes a hierarchic organization of the computational resources building on the concept of an Agent: an autonomous process running on each device that allows executing software in a Function-as-a-Service manner. By natively supporting COMPSs, Colony automatically transforming these functions into task-based workflows being able to exploit the inherent parallelism within the function. Colony agents interact among them to share data and workload enabling the distribution of such workload across the whole Continuum taking advantage of data locality and low-latency network communications to nearby agents. By natively supporting COMPSs, Colony offers a common programming interface to deal with the three computing patterns necessary on IoT-Edge-Cloud services. [102]

6.3.2 funcX

funcX [103] is a distributed Function as a Service (FaaS) platform that enables flexible, scalable, and high performance remote function execution. funcX is designed to support the execution of Scientific workflows over a range of computational resources, from embedded computers to clusters, clouds, and supercomputers, each with distinct access modes. As funcX workloads are often sporadic, resources must be provisioned as needed to reduce costs due to idle resources. funcX build on Parsl to interact with various resources, specify resource-specific requirements, and define rules for automatic scaling.

[43] https://docs.celeryq.dev/en/stable/

7 ML Frameworks

In our pursuit of finding open-source and flexible frameworks with Edge computing capabilities, we have carefully curated a list of libraries that are well-suited for the ICOS ecosystem. These libraries are based on general machine learning open-source frameworks, and have been selected for their versatility and ability to serve a wide range of general purpose use cases. Our selection criteria emphasized the importance of open-source and Edge computing compatibility, allowing for maximum flexibility and integration within the ICOS ecosystem.

7.1 Scikit-learn

Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data pre-processing, model selection, model evaluation, and many other utilities. It contains various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy[44]. Scikit-learn is increasingly used in Edge computing applications to enhance the analysis of data in the IoT side of the Continuum and alleviate the processing burden in the Cloud.

7.2 TensorFlow

TensorFlow is an open-source software library for machine learning and artificial intelligence, that is utilized in various tasks, focusing on training and inference of deep neural networks [104]. TensorFlow is applied in many programming languages, including Python, JavaScript, C++ and Java[45]. Especially for Edge computing, TensorFlow Lite is introduced, comprising a set of tools that allows on-device machine learning by allowing execution of models on mobile, embedded and edge devices. Optimized for on-device machine learning, TensorFlow Lite addresses constraints such as latency, privacy, connectivity, size and power consumption. It supports multiple platforms with high performance, facilitating different ML tasks such as image classification, object detection, pose estimation, question answering and text classification[46].

7.3 TensorFlowLite

A pre-trained model in TensorFlow is converted to a unique format with the help of TensorFlow Lite, an open-source, cross-platform deep learning framework. This format can be optimized for speed or storage. To make the inference at the edge, the special format model can be deployed on edge devices like mobile phones using Android or iOS or Linux-based embedded devices like Raspberry Pi or Microcontrollers.[47]

[44] https://scikit-learn.org/stable/

[45] https://www.tensorflow.org/api_docs/

[46] https://www.tensorflow.org/lite/

[47] https://www.tensorflow.org/lite

7.4 River

River is a library for online machine learning, i.e. machine learning models that can learn on stream data and support different tasks for machine learning tasks, including linear regression, classification, and unsupervised learning, Additionally, it can be used for ad hoc tasks like concept drift detection and computing online metrics.[48]

7.5 PyTorch

PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing. It is an open-source software released under the modified BSD license. PyTorch is released primarily in a Python interface, while operating also under an additional C++ interface [105]. PyTorch provides two high-level features: (a) Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU) and (b) Deep neural networks built on a tape-based automatic differentiation system. PyTorch supports an end-to-end workflow from Python to deployment on mobile operating systems. It extends the PyTorch API to cover common pre-processing and integration tasks needed for incorporating ML in mobile applications, enabling, in this sense, the execution of data analysis in the Edge[49].

7.6 MXNET

Deep learning library suitable for research prototyping and production and allows the integration of Python and support for Scala, Julia, Clojure, Java, C++, R, and Perl for use cases in computer vision, NLP, time series, and more. Finally, it allows distributed training using most of the available hardware: multi-GPU or multi-host training with near-linear scaling efficiency.[50]

7.7 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. Developed with a focus on enabling fast experimentation, Keras reduces developer cognitive load, while allowing the progressive building of simple and advanced workflows for addressing various exercises. By providing industry-strength performance and scalability, Keras can run on TPUs or on large clusters of GPUs, serving the needs for data analytics in the Edge (e.g. running of Keras models in mobile devices)[51].

7.8 dislib

The Distributed Computing Library (dislib) is a Python library provides distributed mathematical and machine learning algorithms (classification, clustering, decomposition, regression, pre-processing, …) through an easy-to-use interface inspired on scikit-learn. The library builds on the PyCOMPSs programming model; hence, it is able to run on any kind heterogeneous distributed infrastructures such as HPC Supercomputers, GPU clusters, clouds and containerized platforms. [52]

[48] https://github.com/online-ml/river

[49] https://pytorch.org/

[50] https://github.com/apache/mxnet

[51] https://keras.io/

[52] https://dislib.bsc.es/

8 Data Pre-processing Operations

All pre-processing operations are designed to improve ML performance metrics. In the following section, we describe some of the common operations that can be performed with the previously mentioned data management frameworks[53]:

Structured Data

8.1 Structured Data

For structured data, the following steps are included[54]:

8.1.1 Data Cleansing

Writing a set of data quality rules for standardization and the eradication of duplicates that are present in the data is a key component of the iterative process of improving data quality. Data cleansing serves this purpose by removing or correcting records from raw data that have damaged or incorrect values, as well as records with a large number of missing columns. This ultimately aids the data quality practitioner by displaying the characteristics of the entities present in the data [106]. The applicable methods include statistical outlier detection, using the mean, standard variation, range, etc., based on the Chevichev theorem; pattern matching, which verifies that the data matches with specific patterns defined for our data; clustering based on Euclidean or other distance to provide support to identify outliers; and data mining techniques [110]

8.1.2 Data Splitting

This includes splitting the data correspondingly on training, validation and testing, including stratification techniques based on the requirements or the framework (the federated learning approach), which may also include stratification techniques, such as random stratified sampling, quota sampling, clustering sampling, etc.

8.1.3 Feature Tunnig

Scaling and normalizing numerical values, impute missing values, clip outliers, and make adjustments to values with skewed distributions are all steps in improving the quality of features in a machine learning model. Scaling and normalizing numeric values involve adjusting the range of values for each numeric feature so that they are on the same scale. This can be important for algorithms that rely on distance measures, such as k-nearest neighbours, as well as for algorithms that use gradient descent, such as neural networks.

Imputing missing values involves replacing missing values with estimates based on the other values in the feature. This can be important because many machine learning algorithms cannot handle missing values and will fail if they are present in the data.

Clipping outliers involves identifying and removing extreme values that are far from the majority of the data. Outliers can have a significant impact on machine learning algorithms, and in some cases, they can even cause the algorithm to fail.

Adjusting values that have skewed distributions involves transforming the values in a feature so that they have a more normal distribution. This can be important because many machine learning algorithms assume that the data is normally distributed, and they may not perform well if the data is heavily skewed.

[53] https://spark.apache.org/docs/latest/ml-features

[54] https://cloud.google.com/architecture/data-preprocessing-for-ml-with-tf-transform-pt1

8.1.4 Feature Transformation

When working with machine learning models, it is often useful to represent data in both numeric and categorical forms. This allows the model to make use of the strengths of both representations and can improve the model's performance.

One way to do this is to use one-hot encoding to translate a categorical feature into a numeric representation. This allows the model to treat the categorical feature as a numeric feature while still maintaining information about the specific categories. Another way to represent both numeric and categorical features is to use learning with counts. In this approach, the categories of a categorical feature are counted, and the count is used as the numeric representation of the feature. This allows the model to make use of the relative frequency of each category, which can be useful in certain types of models [111]

Sparse feature embeddings are another way to represent both numeric and categorical features. In this approach, each category of a categorical feature is represented as a low-dimensional vector, called an embedding. The embeddings capture the relationships between the categories, allowing the model to make use of this information in a more sophisticated way than one-hot encoding or learning with counts.

8.1.5 Feature Selection

Feature selection involves determining the most important features of the dataset while removing redundant features without incurring much loss of information [112]. By removing these features, feature selection can improve the interpretability, complexity, and generalization performance of the predictive model. This can be done in a number of ways, such as using statistical tests to assess the relevance of each feature, using algorithms that search for the best subset of features or using domain knowledge to identify the most important features for the problem at hand or that recognizes those features with highest correlation so it is possible to understand which variables are repeated over the dataset [113].

One benefit of feature selection is that it can help improve the interpretability of the model by reducing the number of features that need to be considered. This can make it easier to understand how the model is making predictions and can help identify the key factors that are driving the predictions.

Another benefit of feature selection is that it can improve the generalization performance of the model on unseen data. By removing irrelevant or redundant features, feature selection can reduce overfitting, which occurs when a model performs well on the training data but poorly on new, unseen data. This can help the model make more accurate predictions on unseen data, which is important for many real-world applications.

8.1.6 Feature Construction

Creating new features is a common technique used in machine learning to improve the performance of a model. Some typical techniques for creating new features include polynomial expansion and feature crossing. Polynomial expansion is a technique in which univariate mathematical functions, such as polynomials, are applied to the existing features to generate new ones. Feature crossing is another technique for creating new features. It involves combining two or more existing features to create a new one that captures their interaction. Features can also be constructed by using business logic from the domain of the machine learning use case. For example, if we are building a model to predict the likelihood of a customer defaulting on a loan, we might create a new feature based on the customer's credit score. This new feature would capture the relationship between the credit score and the likelihood of default and could help the model make more accurate predictions [114].

8.2 Unstructured Data

When working with unstructured data, in the case of images, text or audio, deep learning can be used for feature extraction.

8.2.1 Text

There are several pre-processing techniques that are commonly used on text data for machine learning. Some of these techniques include [115]:

Tokenization: This is the process of splitting a piece of text into individual tokens (i.e. words or phrases).

Stop word removal: This is the process of removing common words such as "a" or "the" from the text, as they are not typically useful for the purposes of most natural language processing tasks.

Stemming and lemmatization: These are techniques for reducing words to their base form (i.e. stem or lemma). For example, the stem of the word "walking" is "walk", and the lemma of the word "was" is "be".

N-gram extraction: This is the process of extracting sequences of N words (i.e. n-grams) from the text. For example, if we set N=2, we would extract all two-word sequences from the text (e.g. "machine learning").

Text normalization: This is the process of transforming words to a single canonical form, such as lowercase, in order to reduce the dimensionality of the text data.

Feature engineering: This is the process of creating new features from the existing text data, such as word counts or tf-idf values, in order to improve the performance of a machine learning model.

8.2.2 Image

There are several pre-processing techniques that are commonly used when working with image data for machine learning. These techniques can help to improve the performance of a machine learning model and make it more effective at analysing and interpreting image data. Some of the most common pre-processing techniques for image data include[55]:

Noise reduction: Removing noise from the image can help to improve the quality of the data and make it easier for a machine learning model to interpret it accurately using techniques such as the median filtering or Gaussian smoothing.

Resizing and cropping: Resizing and cropping the image can help to remove irrelevant information and focus the machine learning model on the most important parts of the image with techniques such as bilinear or bicubic interpolation.

Colour space conversion: Converting the colour space of the image can help to improve the performance of a machine learning model by making the data more amenable to analysis. This can be done using techniques such as RGB to grayscale conversion or RGB to HSV conversion.

Normalization: Normalizing the image data can help to ensure that all of the data is on the same scale and has a similar distribution. This can make it easier for a machine learning model to learn from the data and improve its performance.

8.2.3 Audio

There are several pre-processing techniques that are commonly used when working with audio data for machine learning. These techniques can help to improve the performance of a machine learning model and make it more effective at analysing and interpreting audio data. Some of the most common pre-processing techniques for audio data include [116]:

[55] https://pytorch.org/vision/main/transforms.html

Noise reduction: Removing background noise from the audio signal can make it easier for a machine learning model to focus on the relevant information and improve its performance. This can be done using techniques such as spectral subtraction or wavelet denoising and can be measured using Signal-to-Noise-Ratio (SNR)..

Feature extraction: Extracting relevant features from the raw audio data can help to reduce the dimensionality of the data and make it easier for a machine learning model to work with. This can be done using techniques such as Mel-frequency cepstral coefficients (MFCCs) or spectrograms.

Data augmentation: Augmenting the available audio data can help to improve the generalizability of a machine learning model and make it more effective at handling novel input. This can be done by synthesizing new audio data using techniques such as pitch shifting or time stretching.

Normalization: Normalizing the audio data can help to ensure that all of the data is on the same scale and has a similar distribution. This can make it easier for a machine learning model to learn from the data and improve its performance.

9 MLOps baselines

DVC[56] [Open-source]

Data scientists and machine learning engineers can assure the repeatability of their models using the data version control (DVC) tool. To track and manage the versions of datasets, models, and code, it can handle data pipelines and version them using git to replicate workflows while making effective project management and collaboration by handling huge files and complicated dependencies.

MLFlow[57] [Open-source]

With capabilities like tracking, projects, models, and a model registry, users may design and manage massive data science projects using the open-source MLOps platform MLflow. Although it is cost-free to download, the infrastructure needs to be maintained. MLflow can be integrated with Apache Spark, Azure ML, and AWS SageMaker and is compatible with a number of machine learning libraries on both Python and R. Real-time updates on the progress of the experiment are provided by its main feature, model monitoring.

Kuberflow[58][Open-source]

The Kubeflow project, which was started by Google, aims to make it easier, more portable, and scalable for Kubernetes to install ML models in compliance with the required criteria. Kubernetesis an automated pipeline-based machine learning platform that makes it easier to design, deploy, and monitor ML applications at any stage of their life cycles. A training operator, notebook servers, Kubeflow pipelines, KFServing (a model deployment and serving tools), and an interactive user interface are also included (UI).

MLReef[59][Open-source]

MLReef is an open-source MLOps platform based on git for teams to collaborate and share the results of their machine-learning experiments. This platform manages work in repositories to achieve reproducible, effective, and collaborative ML development.

ONNX[60][Open-source]

ONNX is a universal language for standardizing machine learning models in any framework. It is based on a set of operators, represented as ONNX graphs, for describing common ML operations. This language provides operability, allowing use of any ML framework, and hardware access for optimized performance. ONNX uses protobuf to serialize the graphs and optimize the network. The primary goal is to simplify and facilitate the production pipeline and keep metadata version control, so that the model changes can be tracked at any stage of the serialization.

Kedro[61]

Kedro is a Python framework that offers a solitary environment for creation of trustworthy, reproducible, and maintainable data-driven programs. Its features, which speed up development and reduce the chance of errors through automated testing, code linting, and data validation, include modular code architectures, data engineering pipelines, consistent parameter management, and reusable components. It has a high level of reuse, scalability, and maintainability enables developers to quickly and easily build, test, and deploy applications. It offers a comprehensive set of tools that aid in the creation of efficient, scalable, and simple-to-maintain solutions by developers.

---

[56] https://dvc.org/doc/start/data-management/data-pipelines

[57] https://mlflow.org/

[58] https://www.kubeflow.org/

[59] https://about.mlreef.com/

[60] https://onnx.ai/

[61] https://github.com/quantumblacklabs/kedro

ZenML[62]

ZenML makes it simple for data scientists to put machine learning models into use. It offers enterprise-level scalability and security while automating the MLOps process, including tweaking and testing hyperparameters. It allows data scientists to quickly and effectively create powerful AI applications and supports distributed training tasks on Kubernetes clusters. It is simple to use and a great place to begin with MLOps thanks to its clear user interface.

MLRun[63]

MLRun is an open-source framework for building ML workflows It offers a straightforward and uniform user interface for interacting with different ML libraries, data storage platforms, and cloud environments. It makes it simple for developers to manage and track their experiments as well as construct, test, and deploy ML models. Additionally, MLRun offers capabilities like distributed training, automatic hyperparameter tuning, and versioning for models and data sets. This makes it an effective tool for the creation and application of ML models in a range of settings.

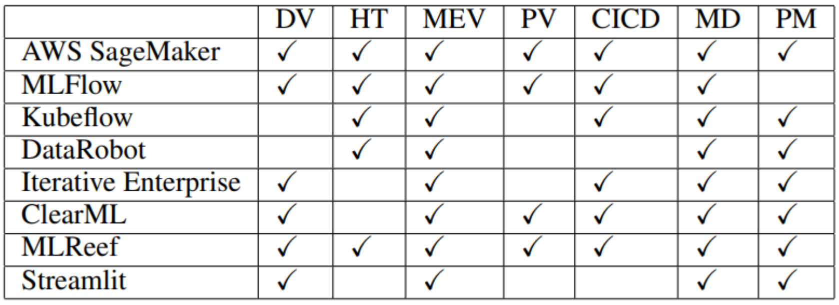
CML[64]

It makes the continuous integration and delivery (CI/CD) procedure for machine learning (ML) applications simpler. For managing ML processes, it offers a command-line interface (CLI) and a Python API. TensorFlow, PyTorch, and scikit-learn are just a few of the widely used ML libraries and frameworks it supports. Additionally, it enables developers to test their pipeline by automatically comparing the pipeline's results to a set of benchmarks and notifying the team of any unexpected changes. It offers a method for packaging the models and dependencies in a repeatable and portable structure that can be used to deploy them in different contexts with little configuration.

Seldon Core[65]

A uniform interface for deploying and scaling models is offered by Seldon Core, an open-source platform created for the implementation of machine learning models on Kubernetes. It offers scalable and fault-tolerant model deployment using Kubernetes and enables the coordination of numerous models to build larger systems. Seldon Core additionally offers tools for monitoring model performance in actual situations and permits the assignment of requests to various models in accordance with established rules or criteria.

With the goal of making a comparison between the different tools for MLOps traceability, the picture below depicts a comparison of some of the most popular frameworks described before:



	DV	HT	MEV	PV	CICD	MD	PM
AWS SageMaker	✓	✓	✓	✓	✓	✓	✓
MLFlow	✓	✓	✓	✓	✓	✓	
Kubeflow		✓	✓		✓	✓	✓
DataRobot		✓	✓			✓	✓
Iterative Enterprise	✓		✓		✓	✓	✓
ClearML	✓		✓	✓	✓	✓	✓
MLReef	✓	✓	✓	✓	✓	✓	✓
Streamlit	✓		✓			✓	✓

Figure 50: Feature[66] comparison of existing platforms for MLOps [11]

[62] https://docs.zenml.io/

[63] https://www.iguazio.com/open-source/mlrun/

[64] https://cml.dev/

[65] https://github.com/SeldonIO/seldon-core

[66] Abbreviations: **DV**: data versioning; **HT**: Hyper-parameter tuning; **MEV**: Experiment versioning; **PV:** Pipeline versioning; **CICD**: Continuous Integration Continuous Delivery ; **MD:** Model deployment

9.1 Real-life Use cases: Experiment Tracking Open-source Libraries

Weights & Biases[67]

W&B is an open-source tool that aids in managing and tracking machine learning experiments for academics and developers. It enables experiment tracking, data versioning, and model management, which enhances cooperation, reproducibility, and version-regression capabilities. It is crucial to remember that W&B is incompatible with other languages and only operates with Python. This makes it a useful tool for Python developers and academics who wish to easily and effectively monitor the performance of their work, data, and models.

Tensorboard[68]

Tensorboard is a TensorFlow visualization tool that enables users to quickly examine and evaluate numerous performance indicators, such as loss and accuracy, while training and assessing machine learning models. In order to better comprehend the behaviour of the model and spot patterns or trends in the data, it also makes it possible to show images, text, and embeddings in a reduced dimensional space. Therefore, Tensorboard is linked with TensorFlow's training and assessment procedures and is a valuable tool for tracking the status of experiments and comprehending the findings of any type of experiment.

[67] https://wandb.ai/site

[68] https://www.tensorflow.org/tensorboard

10 UC1 User scenario – full version

Table 15: User scenarios

LP	Scenari name, user type[69]	Characteristic	Type of data/commands and processing
1.	Transferring to the field(U,R,P)	Robot are prepared, basic maintenance is done. Robot is moved from farm: on wheels (fields in the immediate vicinity) (1) or on transport platform (larger distance, using roads)(2).	1. autonomous, navigation (GNSS+local sensors - avoidance), defined point 2. robot controller,(possible automatic mode to loading/unloading robot from the platform – connectivity robot-platform + information to the panel (status)) **Processing**: EDGE
2.	Mission (common)(U,R)	Map upload, path following, task execution, monitoring, video data collection, navigation, status	Robot storage: vision data, maintenance data, Cloud: Mission monitoring data, status. **Processing**: EDGE, Upload2Cloud
3.	Mission – monitoring/ inspection(R)	Map creation, path following, task execution, video data collection, navigation, status	Robot storage: vision data, maintenance data, Cloud: Mission monitoring data, status, **Processing**: EDGE-EDGE, EDGE-Cloud continuum
4.	Mission - seeding(R)	Map creation, path creation, video data collection, navigation, status	Robot storage: vision data, maintenance data, Cloud: Mission monitoring data, status, **Processing**: EDGE-EDGE, Upload2Cloud
5.	Mission - weeding(R)	Map upload, path following, task execution, video data collection, navigation, status	Robot storage: vision data, maintenance data, Cloud: Mission monitoring data, status, **Processing**: EDGE, Upload2Cloud, Cloud processing
6.	Mission - spraying(R)	Map upload, path following, task execution, video data collection, navigation, status	Robot storage: vision data, maintenance data, Cloud: Mission monitoring data, status, **Processing**: EDGE
7.	Re-filling(R)	Established refuelling/refilling point near transport platform, robot calculates low fuel/fluid level (sensor), abort the mission and calculate the path to drive to the refilling point. After refill back to the previous mission.	Refilling – manually process, Edge: Mission aborting – autonomous: GNSS + local sensors, navigation, Cloud: status, Crop protection refill: ?h, Fertilizer refill: ?h, Refuel: after 8h **Processing**: EDGE
8.	Obstacle detection(R,U)	There is an obstacle in the robot's path that makes it difficult to complete the mission, mission stops, wait for user/system action	Edge: Onboard sensors, Cloud: status, user action monitoring **Processing**: EDGE, Upload2Cloud -> user scenario 14

[69] User type: (U): User; (R): Robot; (P): Transport platform; (A):Admin

LP	Scenari name, user type[69]	Characteristic	Type of data/commands and processing
9.	Predictive maintenance(R)	Before the malfunction (detection), Robot stops	Edge: maintenance data stored, error log stored, Cloud: status, error log, monitoring information **Processing**: EDGE-Cloud continuum
10.	Communication lost(R)	The robot lost communication at field and connect to driving platform. Robot and platform lost connection – store all data.	Robot: LORA+WIFI+xG Platform: LORA+WIFI+xG **Processing**: EDGE-Cloud continuum
11.	Robot 2 platform data exchange(R,{)	After the mission is completed, the robot is loaded onto the platform, and starts sending data to the platform, after which the robot shuts down the system.	Data validation, status, **Processing**: EDGE-EDGE, Upload2Cloud
12.	Data synch at farm (cloud) (P)	The platform will communicate with cloud servers and exchange the data,	Via WIFI, Mission data dump (Maps, validation etc), Video data dump, Maintenance data and logs, Status, **Processing**: Upload2Cloud
13.	Cloud calculations(C)	NN algorithms calculations, etc	**Processing**: Cloud
14.	System update(A)	Process of uploading software/ module updates	Update package from cloud to robot/platform, **Processing**: EDGE-Cloud
15.	ML models improvements(C)	AI/ML models improvements on the cloud	**Processing**: Cloud