This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101069732

aerOS

EUROPEAN IOT-EDGE-CLOUD

D5.1 – Integration, evaluation plan and KPIs definition (1)*

Integration approach and methodology

Deliverable No.	D5.1	Due Date	31-AUG-2023
Type	Report	Dissemination Level	Public (PU)
Version	1.0	WP	WP5
Description	*Document summarising the approach and methodology that the aerOS consortium will adopt for the software integration of the different components of the ecosystem.*		

**The title of this deliverable is suggested to be formally changed.*

Copyright

The aerOS consortium consists of the following 27 partners:

UNIVERSITAT POLITÈCNICA DE VALÈNCIA	ES
NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"	EL
ASOCIACION DE EMPRESAS TECNOLOGICAS INNOVALIA	ES
TTCONTROL GMBH	AT
TTTECH COMPUTERTECHNIK AG (*third linked party*)	AT
SIEMENS AKTIENGESELLSCHAFT	DE
FIWARE FOUNDATION EV	DE
TELEFONICA INVESTIGACION Y DESARROLLO SA	ES
COSMOTE KINITES TILEPIKOINONIES AE	EL
EIGHT BELLS LTD	CY
INQBIT INNOVATIONS SRL	RO
FOGUS INNOVATIONS & SERVICES P.C.	EL
L.M. ERICSSON LIMITED	IE
SYSTEMS RESEARCH INSTITUTE OF THE POLISH ACADEMY OF SCIENCES IBS PAN	PL
ICTFICIAL OY	FI
INFOLYSIS P.C.	EL
PRODEVELOP SL	ES
EUROGATE CONTAINER TERMINAL LIMASSOL LIMITED	CY
TECHNOLOGIKO PANEPISTIMIO KYPROU	CY
DS TECH SRL	IT
GRUPO S 21SEC GESTION SA	ES
JOHN DEERE GMBH & CO. KG*JD	DE
CLOUDFERRO SP ZOO	PL
ELECTRUM SP ZOO	PL
POLITECNICO DI MILANO	IT
MADE SCARL	IT
NAVARRA DE SERVICIOS Y TECNOLOGIAS SA	ES
SWITZERLAND INNOVATION PARK BIEL/BIENNE AG	CH

Disclaimer

Authors

Name	Partner	e-mail
Riccardo Leoni	P19 DST	r.leoni@dstech.it
Saverio Gravina	P19 DST	Saverio.gravina@dstech.it
Ignacio Lacalle Úbeda	P01 UPV	iglaub@upv.es
Michele Mondelli	P19 DST	m.mondelli@dstech.it
Jon Egaña Zubia	P20 S21Sec	jegana@s21sec.com
Harilaos Koumaras Vasilis Pitsilis	P02 NCSRD	koumaras@iit.demokritos.gr vpitsilis@iit.demokritos.gr
Vaios Koumaras	P15 INF	vkoumaras@infolysis.gr

History

Date	Version	Change
19-JUN-2023	0.1	Table of Contents and assignments
14-JUL-2023	0.2	Draft of the Deliverable
21-JUL-2023	0.3	Added contents in all sections
21-JUL-2023	0.4	General fine tuning and internal review
26-JUL-2023	0.5	Internal review (Nikolaos Gkatzios, INF) Internal review (Tarik Zakaria Benmerar, ICT-FI)
28-JUL-2023	0.9	Revised content
01-AUG-2023	1.0	Fine tuning after internal reviews and PC review

Key Data

Keywords	Integration approach, system architecture, continuum
Lead Editor	P19 – DS TECH
Internal Reviewer(s)	P10 ICT-FI, P15 INF

Table of contents

Table of contents 4
List of tables 5
List of figures 5
List of acronyms 5
1. About this document 7
1.1. Deliverable context 7
1.2. The rationale behind the structure 9
1.3. Outcomes of the deliverable 9
1.4. Lesson learnt 10
1.5. Deviation and corrective actions 10
2. Integration strategy and methodology 10
2.1. Adaptive approach 10
2.2. Roles and responsibilities 11
2.3. Timeboxes and iterations 14
2.4. Meetings 15
2.5. Integration Management Platform 19
3. First version of the system Architecture 21
4. Integration tools and integration environments 26
4.1. Integration standard and practices 26
4.1.1. Trunk-based development 26
4.1.2. Automated testing 26
4.1.3. Telemetry-first approach 27
4.1.4. Configuration is code 27
4.1.5. Communications strategy (interoperability strategy) 27
4.2. Integration tools 28
4.2.1. Continuous integration and development tools 28
4.2.2. Potential integration tools analysis 29
4.2.2.1. SonarQube 30
4.2.2.2. OWASP Zap 30
4.2.2.3. Sentry 30
4.2.2.4. Prometheus 30
4.2.2.5. Grafana 31
5. Conclusions 32

List of tables

Table 1. List of acronyms 5
Table 2. Deliverable context 7
Table 3. Meetings planned in each Time Box 15
Table 4. Main features of an integration management platform 19

List of figures

Figure 1. Relationship between WP5 and the rest of the Workplan 8
Figure 2. Adaptive iteration steps 11
Figure 3. aerOS diagram of WPs interaction for integration 12
Figure 4. Correspondence of integration roles with global DevPrivSecOps process 13
Figure 5. Sprint example in Scrum Methodology 14
Figure 6. Flow of the three weeks of Iteration 18
Figure 7. Integration Management Tasks 19
Figure 8. OnlyOffice Logo 19
Figure 9. OnlyOffice features 20
Figure 10: Jira logo 20
Figure 11: Main features of Jira tool 20
Figure 12: Confluence logo 21
Figure 13. aerOS Domain 23
Figure 14. aerOS ecosystem 24
Figure 15. aerOS Management Framework (left: aerOS Management Portal, right: aerOS Federator within aerOS domains) 25
Figure 16. Gitlab CI/CD pipeline composed by jobs and stages 29
Figure 17. GitLab + FluxCD 29

List of acronyms

Table 1. List of acronyms

Acronym	Explanation
DevPrivSecOps	Development Privacy Security Operations
FOM	Federated Orchestration Module
FAI	Frugal AI with Explainability
TSF	Supporting aerOS Features
KPI	Key Performance Indicator
OS	Operating System
IoT	Internet of Things
MVP	Minimum Viable Product
IE	Infrastructure Element
HLO	High-Level Orchestrator

LLO	Low-Level Orchestrator
AI	Artificial Intelligence
API	Application Program Interface
TBD	Trunk Based Development
CI	Continuous Integration
CD	Continuous Development
CaC	Configuration as Code
IaC	Infrastructure as Code
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
RAML	RESTful API Modelling Language
REST	REpresentational State Transfer
SDK	Software Development Kit
DevOps	Development Operations
YAML	Yet Another Markup Language
PromQL	Prometheus Query Language

1. About this document

This deliverable is contextualised in the WP5 of the project aerOS (aerOS integration, use cases deployment and validation). It covers the findings, decisions and advances performed in the first task of the WP in the period M1-M12. The task T5.1 has been running for six months (M7-M12), therefore have generated proper advances that are reported in this document. On the other hand, the second task is focused in use-cases deployment and will be reported in deliverable D5.3 (M18 – February 2024), while the third task (T5.3) is now starting and will be the focus of the team during the next period, to be reported in D5.2 (M24). However, neither of the previous is reported in this document.

The present deliverable describes the approach and methodology for integrating the different software components of the aerOS ecosystem. The ultimate goal of Integration is to create a single and integrated ecosystem so that each software service, module and component responds effectively to the application's functions, requirements and use cases.

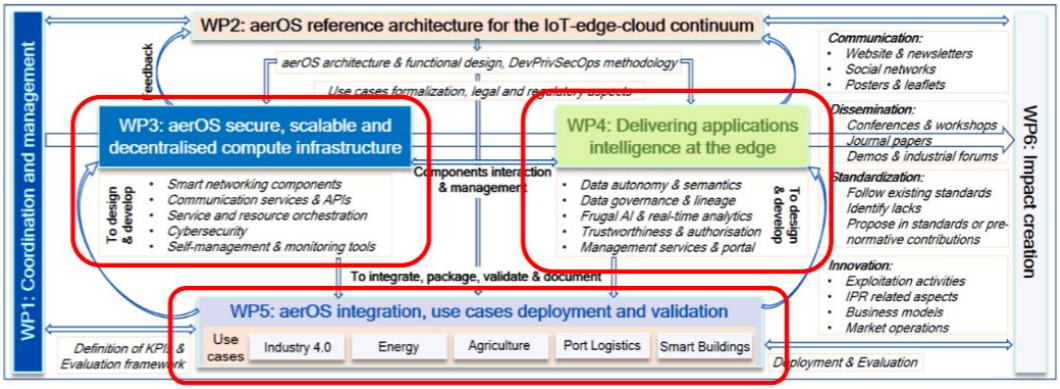
The document is divided into 4 chapters: 1. About this document; 2. Integration strategy and methodology; 3. First release of the system architecture; 4. Integration tools and integration environments. Another aspect that is related to the work performed in WP5 (KPIs definition) will be described in the next deliverable of the series (D5.2).

- In the chapter "About this document", the context of the Deliverable is described, the rationale used for structuring the document, the impacts of the deliverable on the rest of the project, lessons learned, any deviations and corrections made during Task 5.1 to which the deliverable refers.
- The second chapter, "Integration strategy and methodology," describes the software integration approach involved, the roles and responsibilities of the software integration partners, the integration methodology and timeline, and the channels and management tools that will be implemented.
- The third chapter is dedicated to summarizing how the first version of the system architecture will be configured. This chapter is functional in understanding a first Backlog of features to be integrated.
- The fourth chapter, "Integration tools and integration environments", describes integration standards and practices, trunk-based development, automatic testing to validate software, telemetry-first approval, interoperability strategy, and the first integration and development tools that will operate.

1.1. Deliverable context

Table 2. Deliverable context

Item	Description
Objectives	The objectives of this deliverable are: • Analyse the integration requirements; • To describe the integration approach of the aerOS components; • Unify the methodology of the software development; • Harmonize the outcomes of the development activities for producing the final aerOS solution; • Deliver integration guidelines aligned with DevPrivSecOps and technical and packaging documentation, to ensure proper integration. • Define (i) tools, platforms and development framework(s) to be used throughout the integration and (ii) hardware/software APIs to ensure integration with the use case equipment. • Define a continuous integration environment, which will enable system validation, without interrupting the real environment operations.
Work plan	The tasks that represent a background of the D5.1 are: • T2.2 Formalisation of use cases and requirements elicitation [M1-M18] • T2.4 DevPrivSecOps methodology specification [M3-M21] • T3.1 Smart networking for infrastructure element connectivity [M4-M30] • T3.2 Communication services and APIs [M4-M30]

	• T3.3 aerOS service and resource orchestration [M4-M30] • T3.4 Cybersecurity components [M4-M30] • T3.5 Node's self-* and monitoring tools [M4-M30] • T4.1 Data autonomy for homogenisation, semantic interoperability [M4-M30] • T4.2 Data governance, traceability, provenance and lineage policy engine [M4-M30] • T4.5 Trustworthiness, authentication and authorisation [M4-M30] • T4.6 Management services and aerOS management portal [M10-M30] Task 5.1 is a strategic task because it aims to fine-tune development and integration of all aerOS components, developed in WP3-4. Actually, WP3 and WP4 contain the implementation of the overall aerOS components at two levels, the compute infrastructure supported by the smart network component (aerOS Core) and a resource and services orchestrator and the components to deliver intelligence close to the edge (aerOS FAI and aerOS TSF). These developments are provided as inputs to WP5 (particularly, T5.1) that provides integration, deployment of selected use cases evaluation (as described in section 1.2.4.1) and validation procedures targeting both lab-level technology validations and prototyping with demonstrations in the five selected vertical domains. Figure 1. Relationship between WP5 and the rest of the Workplan below shows the relations between WP5 and aerOS' Worplan.  *Figure 1. Relationship between WP5 and the rest of the Workplan*
Milestones	The milestones that represent a background of the D5.1 are: • MS 2 Use cases and requirements defined (M9) • MS 3 Components defined (M2)
Deliverables	The deliverables that represent a background of the D5.1 are: • D2.2 - Use cases manual, requirements, legal and regulatory analysis (1) - (Due Date M9) • D2.4 - DevPrivSecOps methodology specification (1) - (Due Date M9) • D2.6 - aerOS architecture definition (1) - (Due Date M12) • D3.1 - Initial distributed compute infrastructure specification and implementation - (Due Date M12) • D4.1 - Software for delivering intelligence at the edge preliminary release - (Due Date M12)
Risks	The risks that represent a background of the D5.1 are: • Lack on the Technical Requirements and Functional Features • Security and privacy concerns while gathering data for use cases • Data from use cases is not provided in time, or the volume/type of nodes do not fully represent an IoT edge-cloud continuum ecosystem

1.2. The rationale behind the structure

The rationale behind the structure of the Deliverable lies in the need to address two critical aspects of system integration projects: the overall approach and the specific technical elements required for successful integration. The rationale behind the structure of the Report, based on three main contents, can be explained as follows:

1. Integration Strategy and Methodology: The first content, "Integration strategy and methodology," sets the foundation for the entire integration process. It outlines the approach that will be followed to integrate various software components into a cohesive system. By detailing the integration approach, roles, and responsibilities of each software integration partner, the report ensures that all stakeholders understand their roles in the process. This section also covers the integration methodology and timeline, providing a clear roadmap for the integration project's execution. Furthermore, the inclusion of channels and management tools is vital as it allows effective communication and coordination between different parties involved in the integration. By defining the communication channels and selecting appropriate management tools, the report ensures that all stakeholders can stay informed about the integration progress, potential challenges, and any adjustments required throughout the process.
2. System Architecture Configuration: The second content of the report focuses on summarizing how the first version of the system architecture will be configured. This section is essential as it lays out the groundwork for the entire integration process. Understanding the system architecture helps identify potential areas of complexity, potential integration challenges, and critical dependencies. Additionally, by analysing the first version of the system architecture, the report can create a functional understanding of the features that need to be integrated. This information will be valuable when creating a backlog of integration features, prioritizing tasks, and developing a roadmap for the integration process.
3. Integration Tools and Environments: The third content, "Integration tools and integration environments," delves into the technical aspects of the integration process. By describing integration standards and practices, the report ensures consistency and quality throughout the integration. It outlines best practices that will be followed to ensure smooth collaboration between different software components and integration partners. The inclusion of trunk-based development, automatic testing, and telemetry-first approach highlights the importance of maintaining a robust and reliable integration process. Trunk-based development allows for continuous integration, automatic testing ensures that software components function as intended, and telemetry-first approach helps gather valuable data to inform future development decisions.

Moreover, the report discusses a first set of integration and development tools that will be utilized. This is crucial as it ensures that all stakeholders are aware of the tools involved, have the necessary training, and can seamlessly work together in the chosen development and integration environments.

Overall, the combination of chapters on "Adaptive/Agile Strategy and Methodology" and "Integration Tools and Integration Environments" ensures that the system integration approach report covers both the strategic and technical dimensions of the project, fostering a comprehensive understanding of how to approach and execute the integration successfully. It allows stakeholders to embrace flexibility while simultaneously considering the specific technical requirements needed to achieve integration goals effectively.

In summary, the three main contents of the "System Integration Approach" report have been structured to provide a comprehensive understanding of the integration process. The report covers strategic, functional, and technical aspects to ensure successful integration and collaboration among all parties involved. By following this structure, the integration team can approach the project with clarity, efficiency, and a well-defined plan, ultimately leading to a successful integration outcome.

1.3. Outcomes of the deliverable

The main outcomes of this deliverable will mainly focus on:

- Description of the integration approach employed in the aerOS system.
- List of integration tools validated by the consortium.

- Analysis of the potential integration tools that will be used in order to achieve a secure, monitorable, performant, and scalable system.

1.4. Lesson learnt

From the work carried out in Task 5.1 until the completion of Deliverable 5.1 titled "Integration Approach and Methodology," we have learned the following lessons:

- The impossibility of setting up an integration plan without having a more solid global understanding of the selected tools, technologies and perfected requirements. For this reason, it was decided to change the name of the deliverable (from: ***Integration, evaluation plan and KPIs definition*** (1), to: ***Integration approach and methodology***) to present the integration approach and some useful cross-cutting tools. As a matter of fact, this makes sense from a global project approach, therefore this suggestion will be included in the amendment request to the Grant Agreement that will be launched during the next months.
- The definition of the integration approach. Since the requirements and use cases have not been fully defined yet (only a first version of them), the selected option was to bid for an adaptive integration approach rather than a predictive one. This choice was made because the adaptive approach is ideal when the Backlog of User Stories is defined during the implementation of the project and in parallel with its developments.

1.5. Deviation and corrective actions

As described in the DoA, development will follow a two-phase release roadmap, in which an initial version of aerOS-which will assess the functional validity of the platform-will be released in M18, while the final version will be released in M32.

During the first few months of the project, in consultation with the partners responsible for the aerOS system architecture, weekly meetings were held to define an initial version of the architecture and an initial list of technology components needed. In these teleconferences, it was mutually observed that the main deliverables that are inputs for D5.1 (D2.6, D3.1, D4.1), as written in the DoA, were each planned to be released in M12 (August 2023).

The parallelism of the deliverables listed above was the main reason why the scope of D5.1 had to be clarified, becoming a crucial need to start developing an integration plan appropriate to the complexity of the system. Accordingly, this deliverable describes the approach used to carry out Task 5.1. In the second version of this deliverable D5.2 - Integration, evaluation plan and KPIs definition (2), the full integration plan for all aerOS system will be described and developed.

2. Integration strategy and methodology

2.1. Adaptive approach

The basic concept of adaptive life cycles is to embrace change even late in a project life cycle. The main difference with other life cycles is very rapid iterations, lasting from one week to a maximum of four weeks.

Given the complexity of the project and the level of uncertainty, it was therefore decided to proceed with an adaptive approach. Adaptive life cycles require close collaboration between the team and different stakeholders. In this version, the working approach and the next steps required to generate the full integration plan are described.

The adopted strategy to define and agree on an integration plan adjusted to the reality and expected milestones and functionalities of the meta Operating System will be divided into several steps, as illustrated in Figure 2. Adaptive iteration steps.

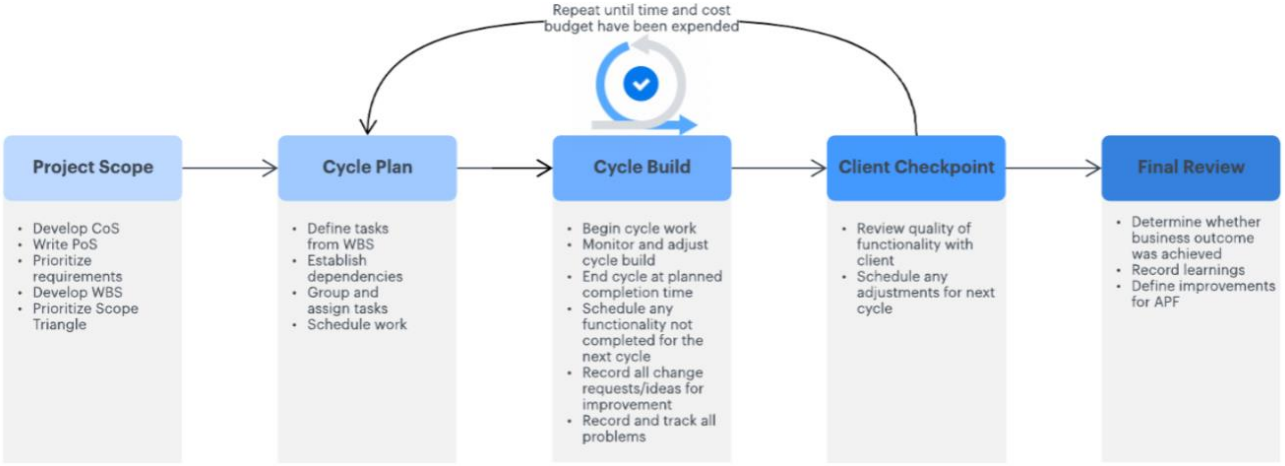


Figure 2. Adaptive iteration steps

In the first phase it has been decided to work in iterations of **three weeks**, which will result in a *delivery* that will be deployed in development environment, to be tested and validated internally. It will also be very important to perform dedicated analysis to capture bottlenecks by visualizing dependencies, that will help to reduce their negative impact on our work.

Following this adaptive methodology, in the coming months, the work will advance for the formal software definition of the Infrastructure Element (IE) precisely and rigorously and aerOS domains that comprise the IoT-edge-cloud continuum. An initial backlog of services and functionalities will be created, that will be validated and prioritized with the help of the necessary stakeholders. Recurrent meetings will be organised to define the work in the different iterations.

As following step, the integration plan will be further described and globally sketched. It is worth mentioning that, following the adaptive approach, dates, goals and duration will be movable (i.e., adaptable) and that the initial integration plan will only be used as global guidance that will be re-visited during the project.

The integration plan will take into consideration the complexity of the system and will provide adaptations and customizations for different pilot scenarios to best integrate all adopted technology solutions. Attached documentation with installation and configuration instructions will be released in the plan to ensure proper integration of the different components in each scenario.

Also, the Integration partners (although focused on technical integration of components coming from WP3 and WP4 in the CI/CD environment), will support the integration of the software outcomes in the pilots, as requested. However, the actual planning and rhythm of those will be offloaded to the proper teams in the task T5.2.

2.2. Roles and responsibilities

By defining the adaptive approach and methodology described above, it is very important to establish clear roles in the whole integration process. Although the plan will be dynamically adjusted to the needs and reality of the project in terms of timing, expected outcomes and cross-dependences, there is the certainty (from WP5 partners) that a governance/functioning structure will remain intact.

Thus, once establishing the roles through this section, it is worth noting that these will remain valid for all the sub-activities related to technical integration of components of the meta–Operating System.

When designing the assignment of roles in the integration process, it is important to bear in mind the following aspects:

- aerOS has two type of integration procedures:

- Integration between technical components of the meta–Operating System, which are the basic services and the aerOS runtime (see the deliverable about architecture – D2.6). Here, the various teams will use the common integration infrastructure to validate the components unitarily and jointly with others as per required. **This integration falls under the task T5.1 and is the only one involved in this deliverable**.
- Integration of aerOS components in the pilots. Here, based on the components required in every defined scenario (see deliverable D2.2), the technical teams together with stakeholders will apply aerOS innovations into their particular use cases. This integration falls under the task T5.2 and is not reported in this deliverable.

- Acknowledging that the work in hand is related to technical components integration, the inner structure of aerOS workplan (Figure 3. aerOS diagram of WPs interaction for integration). It is divided in two work packages, that correspondingly include several tasks which tackle specific aspects of the meta-OS (cybersecurity, data privacy, networking, APIs, etc.). Therefore, whenever deciding the interaction between those, and the integration idiosyncrasy, this will become crucial. In general, the goal has been to create a role structure in the integration aligned with this global workplan so that not to disrupt already created working teams and to maximise efficiency. The next figure depicts the connections of this reflection:

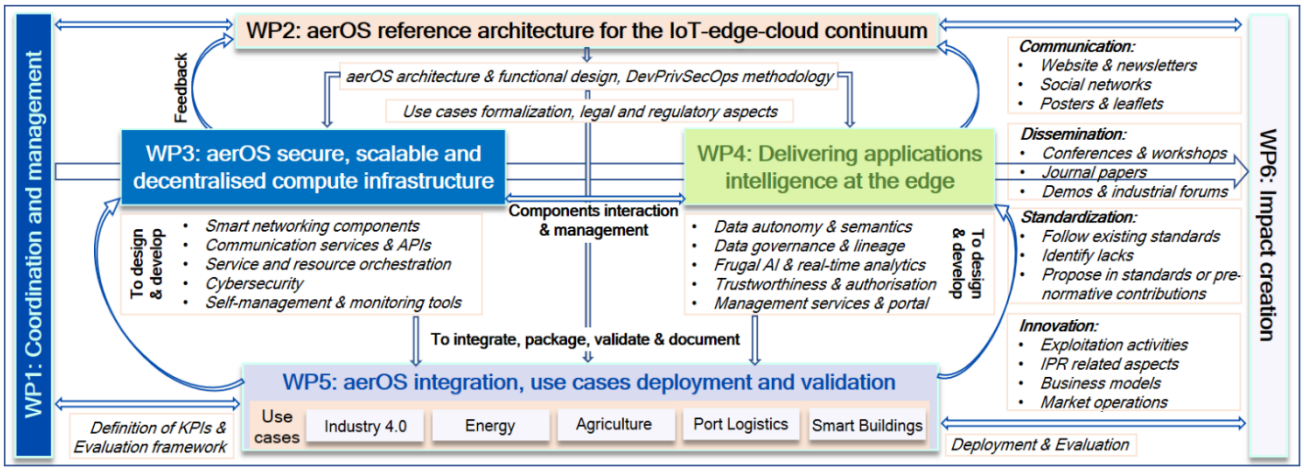


Figure 3. aerOS diagram of WPs interaction for integration

With the previous in mind, the team of T5.1 in aerOS proceeded to define the roles in the Adaptive integration approach. To do that, an analysis was done about the most relevant frameworks that implement Agile methodologies and the roles that are used in those. In particular, the Scrum methodology is characterised by splitting the work (in this case, of integration) in smaller junks, called sprints. Here, the overall concept orbits around a clear, clockwork mechanism that defines the role of Product Owner (the one designing the overall planning and requirements, establishing the prioritization and performing a global overview), the Scrum Master (the one that controls the effective execution of every spring, overseeing the daily process) and the Team Members (that execute the work of each sprint). The Scrum methodology strives for creating short work cycles (of integration, in this case) of about one or two weeks, combined with daily stand-ups for a quick feedback session and status overview. On the other hand, Kanban proposes a methodology with more loosely defined roles but strongly based on individual assignments. There, the different works, or even requirements of a (usually, software) product are defined very granularly and posted onto a Board of tasks. Those, converted into tickets, are appointed to specific members of the team to finalise them in the established time frame. There are several advantages and disadvantages to both, that are extensively analysed in specialized blogs online.

With respect to the integration in aerOS, considering the two bullet points above, both approaches would make sense to some extent. Scrum presents the possibility of a very good self-organization among separated teams, which would fit ideally the Consortium structure of aerOS. In addition, it establishes a strong cooperation structure for cross-functional tasks (e.g., integrating components from different tasks, such as cybersecurity tools with the overall Management Portal of aerOS – see Section 3). On the other hand, the very strict meeting schedule of Scrum might be creating too much overload to teams, hindering the actual goal in a non-for-

commercial-purposes activity in aerOS integration: usability and functionality for advancing in the project. Thus, the integration process could benefit from more fluid working methods.

Therefore, aerOS has decided to adopt its own roles, adjusted to the needs of the research project.

- **Adaptive Approach designer:** This new role is assigned to the person responsible of organizing the chunks of integration in aerOS. It will decide, based on the overall plan outlined in this document and in the status of the project, the timing and global milestones of each of those "integration sprints". In aerOS, this role will be undertaken by **T5.1 leader, partner DSTECH**[1].
- **Product Owner:** Per each of the adaptive integration sprints to be defined in aerOS (inspired on Scrum), the Product Owner will be personalized in two partners: **WP3 leader (SIEMENS)** and **WP4 leader (SRIPAS).** They will be in charge of defining the specific goals of each sprint and also controlling the overall results while solving major or blocking issues.

Here, each sprint will be divided in several sub-sprints, that may correspond to tasks (e.g., T4.5 sprint) or to several tasks, whenever cross-integration will be required (e.g., T3.5-T3.3 integration). The establishment of this division, as well as the inner structure of teams will be up of Product Owners to specify.

- **Team Lead**: They will serve as the head of each organised sub-sprint. Usually, they will be the **leader of one task** and will be in control of the day-to-day integration issues. When more than one task will be involved (foreseeably, most of the cases), one of the task leaders will be nominated as Team Lead (to be decided in an adaptive approach).
- **Team Members** will be composed of the partners (one or more) involved in the different tools/components that are being integrated.

As the integration process in aerOS must be aligned with the DevPrivSecOps methodology, all the principles established there will be followed (see deliverable D2.4). In this regard, apart from sticking to the recommendations, the integration process roles have already been defined bearing in mind the profiles of the users in the GitLab repository of aerOS (serving as the collaborative repository in the project). In Figure 4, there is a short reference between the established relation in this regard.

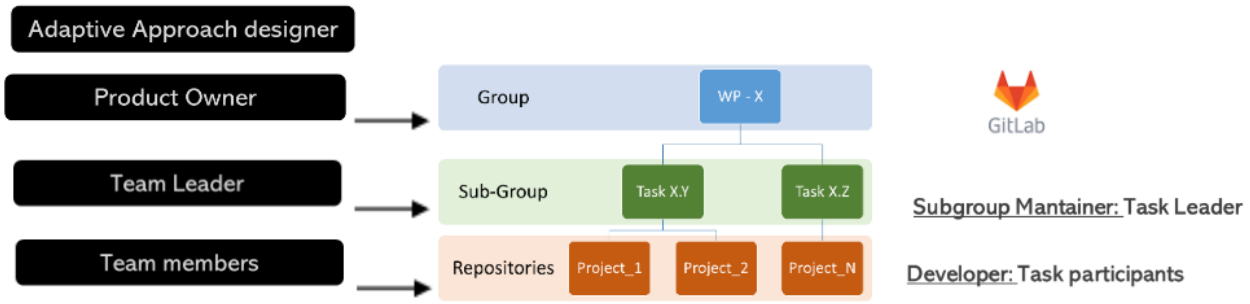


Figure 4. Correspondence of integration roles with global DevPrivSecOps process

2.3. Timeboxes and iterations

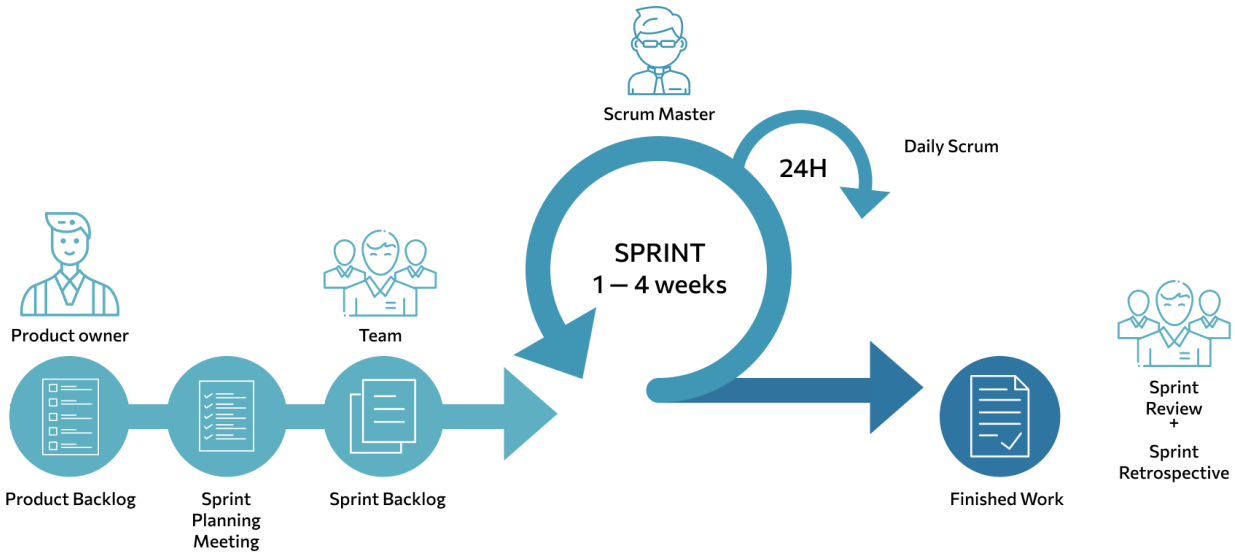


Figure 5. Sprint example in Scrum Methodology

In Adaptive Project Management, time boxes are fixed, predefined periods of time during which specific tasks or activities are performed. The concept of time boxes is closely associated with Agile project management methodologies, such as Scrum. The primary goal of time boxing is to provide a clear structure for work and create a sense of urgency, which helps teams to stay focused and deliver valuable increments of work within short, manageable time frames. Key characteristics of time boxes in Adaptive Project Management:

- Fixed duration: Time boxes have a predetermined length, typically ranging from one week to four weeks, depending on the specific Agile framework being used. As mentioned, three weeks will be the preferred time box in aerOS.
- Invariable scope: The scope of work to be accomplished during a time box remains constant throughout its duration. It helps prevent scope creep and ensures that teams deliver what was initially committed to within the set timeframe.
- Iterative and incremental: Projects progress through a series of time boxes, with each box building upon the previous one. The output of one-time box serves as the input for the next one, allowing for continuous improvement and frequent feedback.
- Regular review and adaptation: At the end of each time box, a review meeting, like a sprint review in Scrum, is held to assess the completed work and gather feedback. The team can then adapt their approach for the upcoming time boxes based on the feedback received.
- Predictability and rhythm: Time boxes help establish a predictable rhythm of work, making it easier for stakeholders to plan and monitor project progress.

The idea behind time boxing is to foster flexibility, continuous learning, and adaptability by allowing teams to regularly reassess their priorities and adjust their approach based on changing requirements and feedback. This approach is particularly beneficial in complex and uncertain projects, where requirements may evolve, and having the ability to adapt quickly is crucial for success.

Overall, time boxes in Adaptive Project Management enable teams to maintain a sustainable pace, deliver valuable increments of work, and continuously improve their processes throughout the project's duration.

Because of adaptive methods break activities down into small increments with minimal planning and do not directly involve long-term planning. For this task, given the complexity of the aerOS system to be implemented and the large number of partners involved, it was proposed to adopt time-boxes of three weeks and to collaborate actively it was agreed to hold stand-up meetings every week in order to check development progress.

In addition, at the end of each iteration, a meeting will be held to organize the next iteration (User Story Refinement) and a retrospective will be done to analyse and discuss the completed iteration.

Finally, we must consider the incremental value that this approach will bring to aerOS ecosystem development. In the context of Agile software development, the Sprint is a time-boxed iteration (Figure 5. Sprint example in Scrum Methodology), typically lasting one to four weeks, during which a specific set of features or user stories is developed and made potentially shippable. The incremental value refers to the tangible and usable value delivered to the stakeholders at the end of each Sprint. Here's a breakdown of the incremental value in a Sprint Agile approach:

- **Iterative Development:** Agile teams work in short iterations (Sprints), where they build small, functional increments of a product during each cycle. These increments are built upon the previous ones, gradually adding more features and improvements. Each Sprint delivers a potentially shippable product, even if it doesn't contain all the planned features.
- **Early and Continuous Delivery of Value:** Unlike traditional waterfall development, where the final product is delivered at the end of the entire development cycle, Agile development aims to deliver value early and continuously. With each Sprint, the product grows, and stakeholders can start using and benefiting from the delivered features much sooner.
- **Feedback and Adaptation:** Frequent delivery of increments allows stakeholders to provide feedback and adjust during the development process. This feedback loop is critical for refining the product and ensuring that it aligns with the stakeholders' needs and expectations.
- **Reduced Risk and Increased Transparency:** Incremental delivery mitigates the risk of delivering a product that doesn't meet the stakeholders' requirements or market demands. As the product evolves incrementally, issues and potential bottlenecks become visible earlier, enabling the team to address them promptly.
- **Flexibility and Adaptability**: Agile methodologies embrace change, and the incremental value approach reinforces this flexibility. As new insights emerge or priorities shift, the team can adjust the upcoming increments, accordingly, ensuring that the most valuable features are always prioritized.
- **Prioritization and Focus on Value:** The team focuses on delivering the most valuable features first, which means that stakeholders can see the benefits of the product early on. This approach allows the product owner and stakeholders to make informed decisions about the product's direction based on the delivered increments and their actual value.
- **Continuous Improvement**: With each Sprint, the team learns from their experiences and applies improvements in subsequent iterations. This continuous improvement process helps optimize development efficiency and the value delivered over time.

In summary, adopting this approach aerOS will get incremental value means delivering working increments of the product at the end of each Sprint, providing stakeholders with tangible value early and continuously, and enabling iterative feedback and adaptability to maximize the product's overall value and quality.

2.4. Meetings

In an Adaptive Integration approach Time Box (Iteration), meetings will take place according to the following Table 3. Meetings planned in each Time Box:

Table 3. Meetings planned in each Time Box

Meeting	Purpose	Inputs	Outputs
Sprint Planning Meeting	The Sprint Planning Meeting marks the beginning of a Time Box (also known as a Sprint) and serves to define the scope of work for that specific iteration.	**Product Backlog:** A prioritized list of user stories and other work items that represent the requirements and features to be implemented in the	**Sprint Goal:** The team establishes a clear and concise objective for the Sprint, outlining what they aim to achieve during the Time Box. **Sprint Backlog:** A

		project. **Definition of Done**: The agreed-upon criteria that define when a user story or task is considered completed and meets the required quality standards.	selection of user stories and tasks from the Product Backlog that the team commits to completing in the Sprint. **Plan**: The team creates a plan outlining how they will approach the selected user stories and tasks to meet the Sprint Goal.
Standup Meeting	The Standup Meeting is a short, daily synchronization meeting that helps the team stay aligned, identify potential obstacles, and maintain focus on their Sprint Goal.	**Sprint Goal:** The objective set for the current Sprint during the Sprint Planning Meeting. **Sprint Backlog:** The list of user stories and tasks committed to during Sprint Planning. **Impediments:** Any challenges or roadblocks that team members may be facing.	**Updated Task Status:** Each team member shares what they worked on the previous day, what they plan to work on that day, and if they encountered any impediments. This information helps the team understand progress and identify any issues that need attention.
User Story Refinement Meeting (also known as Backlog Refinement or Grooming)	The User Story Refinement Meeting allows the team to review and clarify upcoming user stories to ensure they are well-prepared for future Sprints. It is relevant to clarify that users will refer in aerOS to technical members expecting a functionality based on project technical requirements.	**Product Backlog**: The list of user stories and other items that are not yet scheduled for a specific Sprint.	**Refined User Stories**: User stories that have been reviewed, clarified, and broken down into smaller tasks, making them ready for selection and implementation in future Sprints.
Sprint Review	The Sprint Review occurs at the end of each Sprint and involves the team presenting the work completed during the Time Box to stakeholders.	**Sprint Goal:** The objective set for the current Sprint during the Sprint Planning Meeting. **Completed User Stories:** The user stories and tasks that the team has finished during the Sprint. **Feedback and Observations:** Input received from stakeholders during the Sprint Review.	**Feedback and Insights:** Stakeholder feedback and observations about the completed work, which can influence future iterations and help refine the product backlog.

Sprint Retrospective Meeting	The Sprint Retrospective Meeting is held at the end of each Sprint to reflect on the team's processes and identify areas for improvement.	**Sprint Goal:** The objective set for the current Sprint during the Sprint Planning Meeting. **Completed User Stories**: The user stories and tasks that the team has finished during the Sprint. **Feedback and Observations:** Input received from stakeholders during the Sprint Review. **Team Insights:** Observations and suggestions from team members about the Sprint's processes and collaboration.	**Action Items:** Identified improvements and changes the team commits to making in the next Sprint to enhance their productivity, collaboration, and overall performance.

At this point, it has been decided to propose a series of 3-week time-boxes. In the next phase of the project, there will be in-depth discussion to properly define the appropriate duration of a time box and evaluate the feasibility of the proposal. The meetings previously depicted will take place in the following manner:

And so, the three-week Time Box comes to an end, culminating in a successful and collaborative Agile sprint (Figure 6**).** The team can now use the insights from the retrospective to refine their approach, ensuring that they continuously deliver value and adapt to meet the project's changing needs.

Week 1

On Monday morning, the team gathers for the Sprint Planning Meeting. They start by reviewing the project's overall goals and priorities with the Product Owner. Together, they discuss the user stories in the product backlog and determine which ones should be worked on during this Time Box. After careful deliberation, the team commits to a set of user stories that they believe they can complete within the three-week period.

Week 2

As the second week begins, the team continues with their work. The team meets on Wednesday for a Standup Meeting. During this 30-minute meeting, each team member shares what they worked on the previous day, what they plan to do today, and if they have any challenges or impediments. This quick check-in helps the team stay aligned and ensures everyone is on the same page.

Week 3

- At the start of the third week, the team carries on with their daily standups on Monday and Tuesday. The frequent check-ins have been beneficial, as any obstacles encountered were promptly addressed, keeping the team's progress on track.
- On Wednesday, the team gathers for the Sprint Review. They showcase the work they have completed during the Time Box to the stakeholders, demonstrating new features and functionalities. The Product Owner provides feedback, and any necessary adjustments are discussed to ensure that the project is meeting its objectives.
- Thursday brings another daily standup, providing the team with an opportunity to fine-tune their work for the remaining days.
- Finally, on Friday, the team concludes the Time Box with the Sprint Retrospective Meeting. They reflect on the past three weeks, discussing what went well and identifying areas for improvement. The team collaborates on action items to enhance their processes and practices for the next Time Box.

Figure 6. Flow of the three weeks of Iteration

2.5. Integration Management Platform

Integration Management Platforms are necessary for aerOS integration to facilitate collaboration, organization, and communication among team members. As illustrated in Figure 7. Integration Management Tasks, there have been selected two type of Integration task:

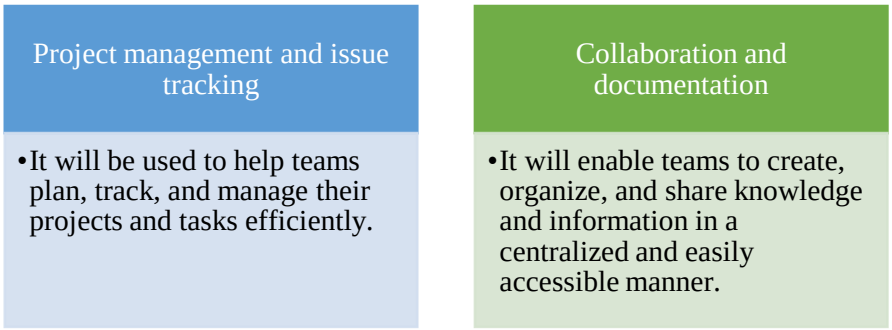


Figure 7. Integration Management Tasks

Each platform serves different purposes, but they often complement each other when used together. Table 4 lists the main features of the platform:

Table 4. Main features of an integration management platform

Issue Tracking	These platforms provide robust issue tracking capabilities, allowing teams to create, manage, and prioritize tasks, bugs, and other project-related issues.
Project Management	They support various project management methodologies, such as Agile, Scrum, and Kanban. Teams can use boards, backlogs, and customizable workflows to manage projects efficiently.
Collaboration	Integration Management Platforms foster collaboration among team members, enabling real-time communication, commenting, and notifications.
Documentation	Users can create and share rich documentation, including project requirements, design specifications, meeting notes, and more.
Knowledge Base	These platforms serve as a centralized knowledge base where teams can store and access important information, making it easy to find and share knowledge
Customization	Users have the flexibility to customize the platform to fit their specific needs by setting up custom fields, workflows, and permissions.
Reporting and Dashboards	They offer built-in reporting features to monitor project progress, track performance metrics, and gain insights into team productivity. Customizable dashboards provide real-time project data.
Integration	Integration Management Platforms often integrate with other tools commonly used in the software development process, such as version control systems, continuous integration servers, and test management tools.

By leveraging these features and functionalities, teams can streamline their project management processes, enhance collaboration, and maintain well-organized documentation and knowledge sharing.

Figure 8. OnlyOffice Logo

In order to achieve the listed requirements and face up the different platforms in a whole tool, a potential considered candidate might be OnlyOffice (Figure 8. OnlyOffice Logo). OnlyOffice is a comprehensive office suite that provides a set of productivity tools for creating, editing, and collaborating on various types of documents. It includes word processing, spreadsheet, and presentation applications similar to other office suites like Microsoft Office or Google Workspace (formerly G Suite). However, it is not optimised to handle Agile-like activities and might imply unnecessary use of resources (online repository is not necessary).

Key features of OnlyOffice include, as showed in Figure 9. OnlyOffice features**:**

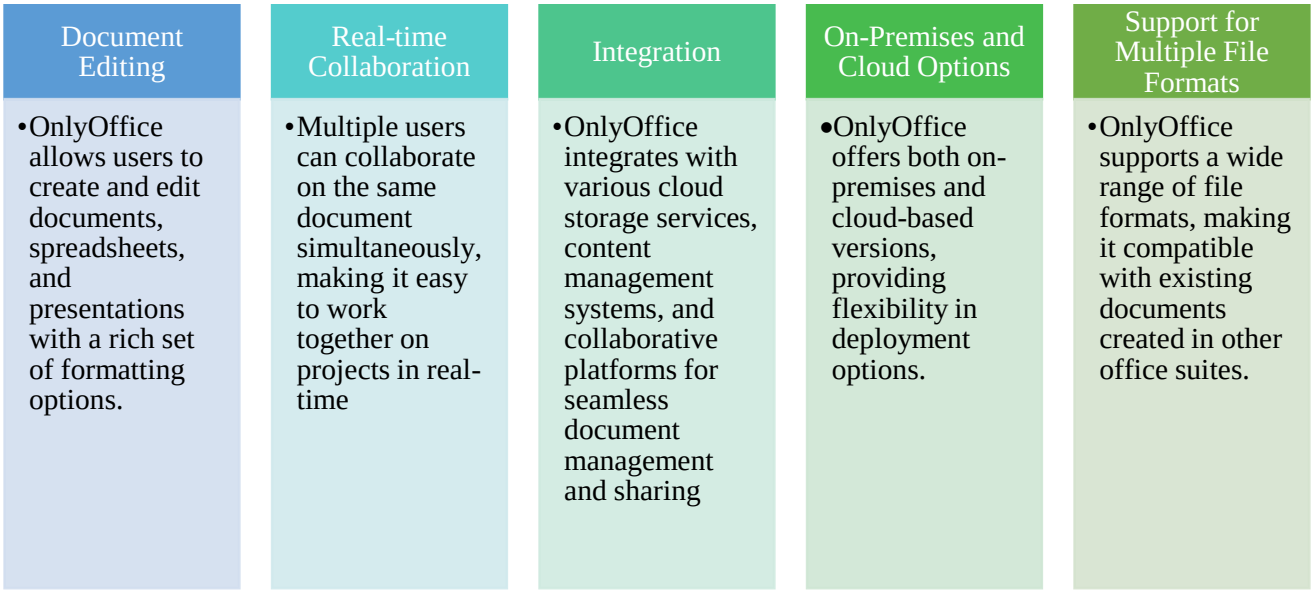


Figure 9. OnlyOffice features

Jira (Figure 10**)** is a popular project management and issue tracking tool developed by Atlassian. It is widely used by software development teams, as well as other industries, to plan, track, and manage projects and tasks efficiently. As relevant advantages, it can well connect to CI/CD and other management tools, and has a long trajectory of usages in many sectors, in both commercial and research activities.

Figure 10: Jira logo

Key features of Jira (Figure 11: Main features of Jira tool) include:

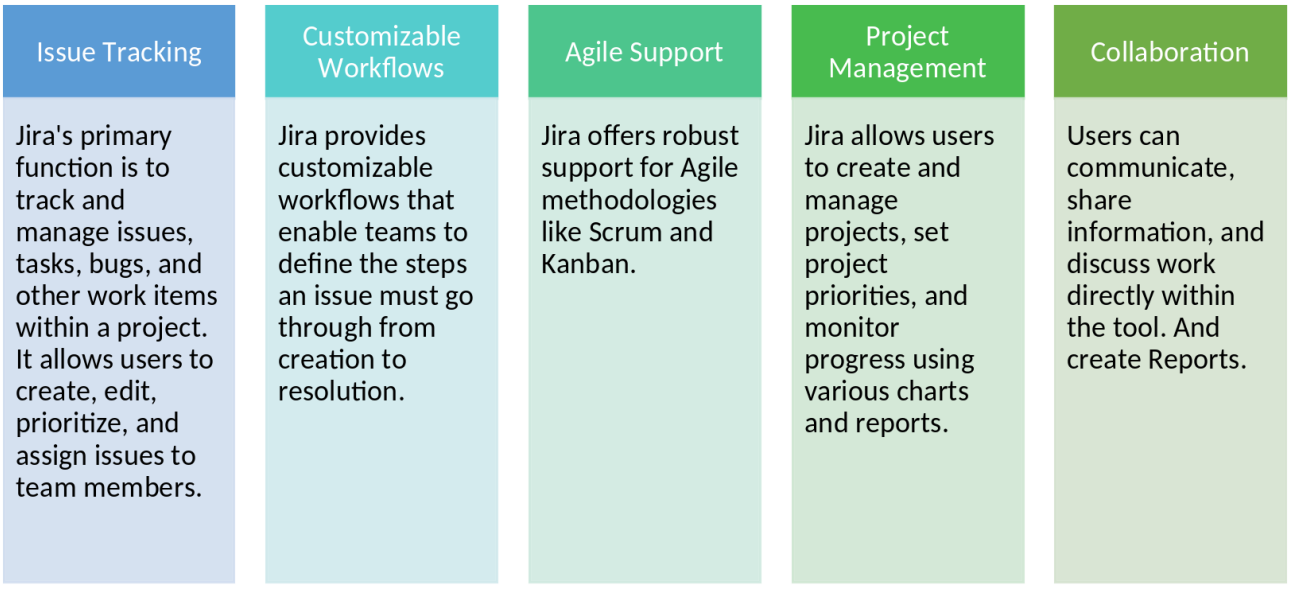


Figure 11: Main features of Jira tool

Jira integrates with a wide range of other tools and services, such as version control systems (e.g., Git), CI/CD tools, test management tools, and more. This ensures seamless flow of information across different tools in the development process. One of these tools is Confluence (Figure 12: Confluence logo), a popular team collaboration and documentation tool developed by Atlassian. It is designed to help teams organise, share, and collaborate on content and knowledge within an organisation. Confluence organises content into Spaces, which act as containers for related pages and information. Teams can create spaces for different projects, departments, or topics. Pages are the building blocks of Confluence. Users can create, edit, and format pages with a user-friendly editor that supports rich content like text, images, tables, and macros. Confluence automatically tracks versions of pages, allowing users to view previous versions, compare changes, and restore earlier states if needed. Users can label pages with tags to categorise and organise content. Labels help in content discovery and navigation.

Figure 12: Confluence logo

3. First version of the system Architecture

As of today, numerous computing systems, private clouds or stand-alone processing units with heterogeneous resources capabilities and access to a limited set of available IoT services, are scattered across the web from far edge installations to cloud premises. This stands as a hinder to resource owners, from various industry verticals, when they try to address the growing needs of IoT deployments within their organizations. The same problem is also present for daily life emerging IoT use-cases where many computing units and private networks primarily function as data concentrators and forwarding equipment, sending vast amounts of data to centralize commercial cloud infrastructures, which provide large computing and storage capabilities which can serve users' needs. These cloud and centrally located services are controlled by a select group of service providers and thus, unfortunately, this approach prevents vertical IoT stakeholders from having full control over their services and data governance. Moreover, while a variety of services have been developed for different industry verticals, they cannot be easily reused. Each time a new organization seeks to solve similar problems, they must either develop solutions from scratch or extensively adapt their existing runtime environments. This lack of a common foundation for IoT service developers hinders the efficient utilization of existing solutions for similar needs. **aerOS** envisions and designs an **architecture -** more precisely has already provided an initial design blueprint and just started to materialise minimal valuable products (MVP) deployments -, of a **meta-OS** to solve this issue. Such meta OS which provides, orchestrates and finally relies on the integration and coordination of a **network and compute fabric** and a **service fabric** to create a federated environment where computing resources are transparently accessible, services are efficiently deployed and managed, and IoT service developers can leverage existing solutions for a wide range of applications, even if they have a limited ownership on computing and service resources. Thus, aerOS is designing an architecture to establish a unified execution environment for IoT service developers across a distributed computing landscape, encompassing diverse geographical and administrative domains with varying compute and networking capabilities devices and over several operating systems. A leading idea within the **aerOS architecture** is its alignment with two recent data management approaches, namely, **data mesh** and **data fabric** in order to achieve a holistic view of all the resources available and their respective status and thus exploit this capability. This will help to maximise usage of distributed resources under a **federated orchestration** process in order to achieve most efficient placement of requested IoT services all over the edge to cloud path. The IoT-Edge-Cloud continuum introduces a highly distributed and dynamic data landscape. aerOS thus ensures the goal of providing a holistic view of the data available in the IoT-Edge-Cloud continuum, while enabling data governance policies that can ensure a responsible use of data.

The establishment of a network, compute and service fabric provides the basis for **federation** which supports collaboration, resource sharing, workload distribution, and interoperability across multiple administrative domains within the IoT edge-to-cloud environment, even if these domains represent different organizations, companies, or stakeholders that contribute computing resources and services to the overall ecosystem. On top of this federated resources environment, **orchestration** shines providing the capability for orchestration of logical resources and service chain execution beyond local domain, spanning along the continuum through an overall view of it and thus offloading requests to other domains as needed. Thus, aerOS orchestration is purposed to solving constraint-based double optimization problems, with data about application requirements and infrastructure as input.

aerOS architecture, foresees the decision-making process for orchestrating services to be decentralized and split into two modules: the **High-Level Orchestrator (HLO)** and the **Low-Level Orchestrator (LLO)**. These modules are designed to imbue aerOS with intelligence and flexibility when allocating computing resources for the containerised workloads within the continuum. Advanced AI algorithms will be employed to optimise parameters like latency, enabling smart decisions on which resources to utilize within the continuum. To ensure accurate decision-making, aerOS architecture will apply such intelligence to overcome the restrictions imposed by locality. This is achieved by extending visibility across the entire continuum and is facilitated by the federated infrastructure overview, offered by a distributed network of interconnected **brokers**, which allows for observation of the continuum's current state any time at any place. The component responsible for connecting these different domains is the **aerOS Federator**, and when combined with the **HLO** and **LLO,** it forms the **aerOS Federated Orchestrator**.

The basic units that aerOS architecture employs as resources to federate and orchestrate, having abstracted underlying hardware and Operating Systems, are:

- **Infrastructure Element:** the most granular entity of computing (able to execute workloads) in the continuum. It can take many forms.
- **Domain**: Grouping of Infrastructure Elements according to certain aspects defined by aerOS.
- **Data Fabric (federation):** The conception of all data available in a continuum as a single box that can be queried and will forward the proper information. Mechanisms within are rather sophisticated.

Having described the dominant role of **Federation and Orchestration** for aerOS continuum efficient and productive function, it is obvious that aerOS architecture could not just employ these concepts without extending on already existing ideas and techniques. It is important for the global Integration plan of the project to briefly present aerOS view and implementation choices on these two ideas, as they are core components of aerOS architecture, and guided important decisions regarding components that should be deployed within each of the aerOS domains and how they should be interconnected.

The data fabric stands in the heart of **aerOS Federation** as a metadata-driven architecture that automates the integration of data from heterogeneous sources and exposes these data through a standard interface. The Data Fabric is responsible to consume raw data from each domain and IE and transform these to a common, and interpretable information model across all aerOS consumers, which can be provided all over the continuum, from edge to cloud, as needed. The transparent and automated information exchange and fusion is based on NGSI-LD protocol which implements the concept of a "**distributed state repository**" envisioned within aerOS architecture. In the context of the aerOS architecture, a distributed state repository is designed as a decentralised storage system responsible for maintaining the state information regarding the different elements within every domain. **Context Brokers** are the implementing components of this architectural block and have the capacity of exchanging contextual information based on established mechanisms following the widely adopted standard NGSI-LD. Thus consumers, for example other aerOS domains orchestrators or trust agents may just ask for data and get it in return without having to know or query themselves the specific aerOS domains or IE located anywhere along the path. A closely related aerOS architecture decision is the use of **knowledge graph** for data fabric to represent and expose underlying data. **Knowledge graph**, which is a promising technology for the realization of the Data Fabric architecture, enables representing concepts that relate to other concepts, by semantically annotating data through ontologies. aerOS domain may include multiple IEs, where new data sources and data consumers dynamically become part of the distributed knowledge graph. In this architecture, **Infrastructure Elements** (IEs) will group into domains, each of which will incorporate, at least, one **Context Broker**, which operates as a **Context Registry**, responsible for storing information about the available data types accessible through other Context Brokers within the aerOS domain. Whenever a data-providing domain registers a new data product in its local Data Fabric, all other Context Brokers may update (via customised registrations) their respective Context Registries to include this new information. Consequently, when a data-consuming domain requests a data product served by other Context Brokers, the local Context Broker is aware of the neighbouring Context Broker from which the data product can be obtained. Utilising this knowledge, the local Context Broker communicates with the neighbouring Context Broker and retrieves the data product on behalf of the consumer. As described earlier, a Federator component is part of every aerOS domain and encompasses Domain Registry and Domains Discovery capabilities. Thus, each IE will provide a Context provider facility, each domain encompasses a federator component capable to register and discover other

domains and also capable of exposing domain's status, i.e., a combined information of domain's capabilities availability.

Being aware of the process chosen to establish a federated environment across aerOS where information is transparently available to all consumers in an interpretable and efficiently retrievable way, the focus can be put again on the orchestration process which takes place within each aerOS domain - that has the "luxury" of knowing and addressing, for deployment requests, all other domains. As also mentioned above aerOS Orchestration is designed to be implemented in two layers: the **HLO** and the **LLO**. By design, every aerOS domain has one HLO instance and this component interfaces with other aerOS services (e.g., , AI decision support modules or the trust manager module), which may offer intelligent support for the final deployment decision. This intelligent support is based both on the knowledge of the continuum (aerOS federator provides this for free) and the seamless integration of AI services. Thus, user requests for IoT services deployment are best served as HLO can make the most efficient decisions regarding the targeted placement of services to selected aerOS domains or IEs. Following this decision HLO provides a Decision *Blueprint* to the specific Low-level Orchestrators to schedule the actual placement to the underlying computing resources. The LLO knows how and where (within the compute and network fabric) to address the actual resources and send the deployment requests to the container-based runtime environment as appropriate. It is important to say here that based on the aerOS architecture, the interface for the communication among HLO and AI and trust tools is designed to comply to one and only API description so that each aerOS domain operator might choose to deploy their AI module which might enforce a targeted or more efficient placement policy per domain.

The above-described concepts are at the heart of aerOS innovation, and the implementing components framed with security and privacy mechanisms establish the core of each aerOS domain.

As said before the main constituents of aerOS are:

- **Infrastructure Element** which is the fundamental building block of the aerOS system providing the computational infrastructure necessary to deploy and manage workloads and it can be any physical or virtual entity that supports containerized workloads. It provides network connectivity and storage capacity and can expose its state under a well-defined API.
- **aerOS domain** which is composed of one or more IE and is a complete aerOS domain hosting and sharing among all its IEs aerOS basic services. Among these basic services are the data fabric provisioning, orchestration and federation services as described before, security services, networking facilities. Figure 13 below provides a schematic overview of an aerOS domain.

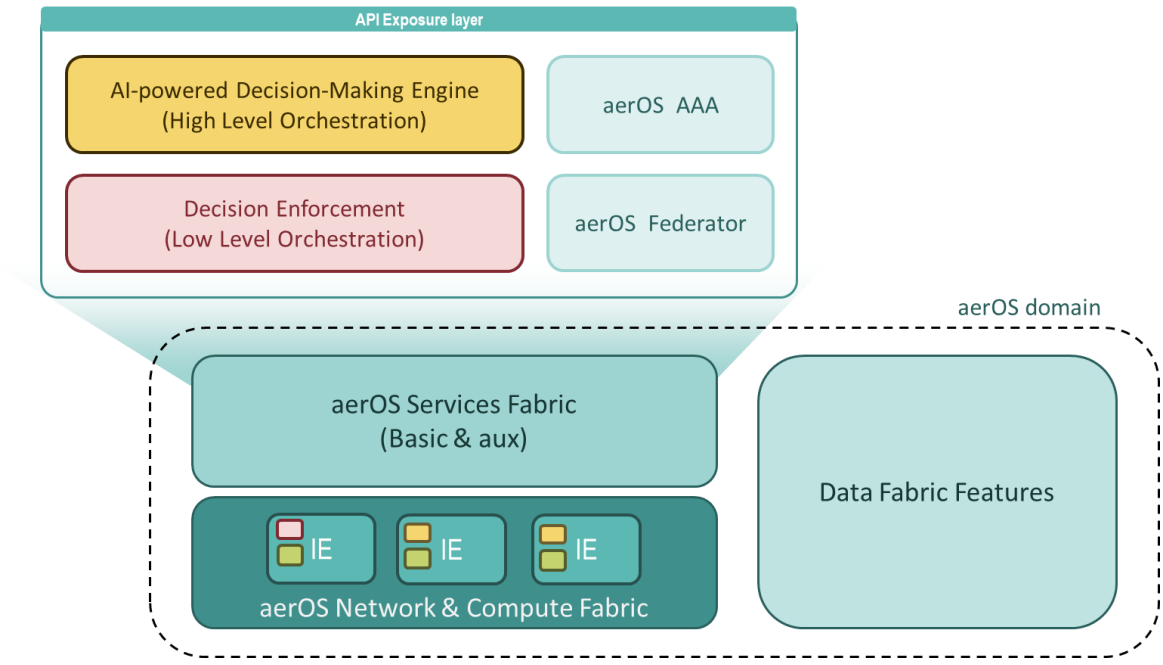


Figure 13. aerOS Domain

Having designed the main services and respective functionalities within and across aerOS domains, an **aerOS Management Portal** is envisioned to integrate the whole picture and provide bindings to users', controlled, access to the system, constituting a single window for managing aerOS meta-OS. Positioning the aerOS Management Portal in the continuum, it will live in one of the domains (selected by the deployer / owner of the continuum). Additional, singleton, aerOS services deployed on it, are following all the main principles of aerOS architecture (containerised format, capable to deploy on IEs etc), and can easily be migrated to some other aerOS domain should a failure exist or for some reasons administrators choose to do so. In this case the whole aerOS meta-OS will continue to properly execute as before, with one single entry-point which has just been mitigated to another domain.

This aerOS entry-point hosts components that provide singleton services across all aerOS domains but, as said before, can be mitigated to another domain and thus have another aerOS entry-point. These components are:

- **Access to User and Policies registry**, holding thus the heart of aerOS AAA system. This will not necessarily be deployed where the portal is, but it must ensure proper connection to the data for guaranteeing access (users, roles, permissions, etc.).
- **aerOS dashboard** which is the graphical entry-point for users wishing to access aerOS by either registering new aerOS domains or requesting IoT service deployments.
- **Entry-point balancer** which is a configurable component and makes the best choice, based on user's request and system availability, regarding which aerOS domain HLO should handle this deployment request.

In the following Figure 14. aerOS ecosystem, a scenario regarding the connectivity of the entry-point domain with the rest of the aerOS domains is presented.

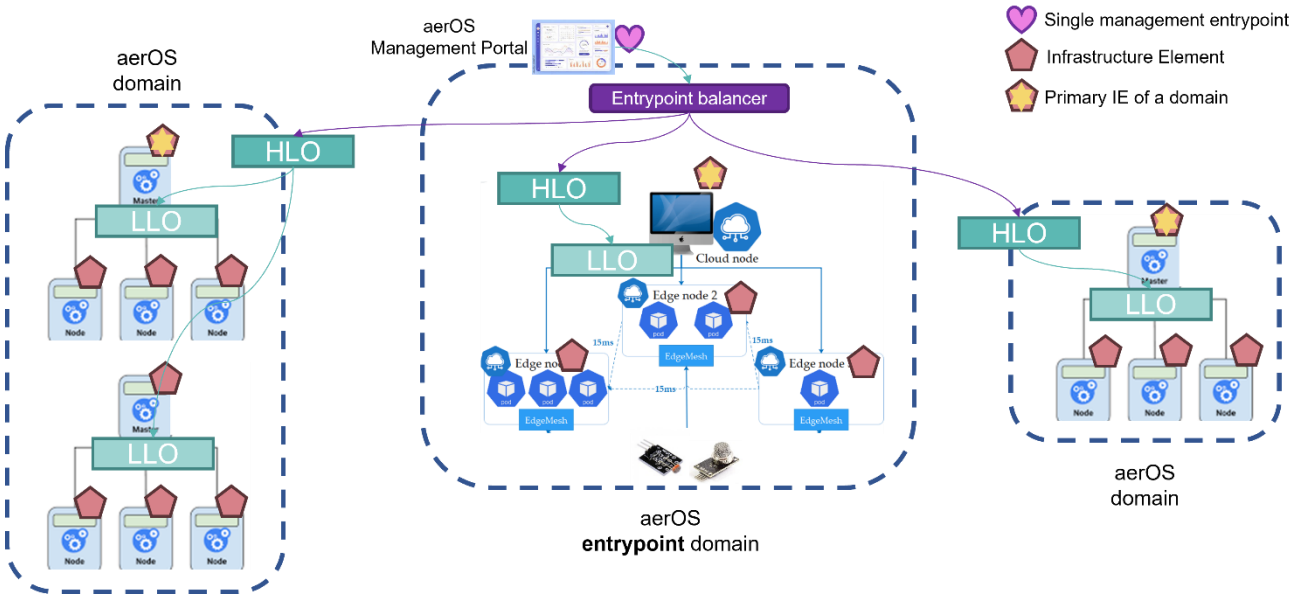


Figure 14. aerOS ecosystem

As explained, aerOS architecture has chosen standardised information models to be responsible for every data exchange across all aerOS data producers and consumers. Thus, the Data as a Product concept will be handled in aerOS, which allows the data to be produced, transformed, and exchanged based on well-known models so that they can be interpretable by every component that would need to make use of it. A similar approach is used for the flow of user intentions towards implemented IoT services. aerOS orchestrator is expecting user intentions based on templated requests' expressions, *Intention Blueprints*, which will lead to intelligent-supported decisions and to enable the HLO to produce *Decision Blueprints* to subsequently feed the LLO. This decision support structure implies an abstraction between user descriptions and decision mechanisms enforcement, allowing thus for different implementations for common components, should future users wish to do so.

Beyond the IoT services deployment requests, aerOS portal has an equally important role in the management of the aerOS meta-OS. It is not the single entity regarding aerOS management but is part of the **aerOS Management Framework.** This management framework includes components of the aerOS management

portal, as the User and policies registry, a policy enforcement point, initial entry-point for aerOS domains registration, dashboard space for management activities. However, **aerOS Management Framework** (Figure 15. aerOS Management Framework (left: aerOS Management Portal, right: aerOS Federator within aerOS domains), goes far beyond these and includes mechanisms and common components distributed across all aerOS domains (aerOS federator component responsibility) that oversee the creation and maintenance of federation among all domains, establishing thus the aerOS continuum.

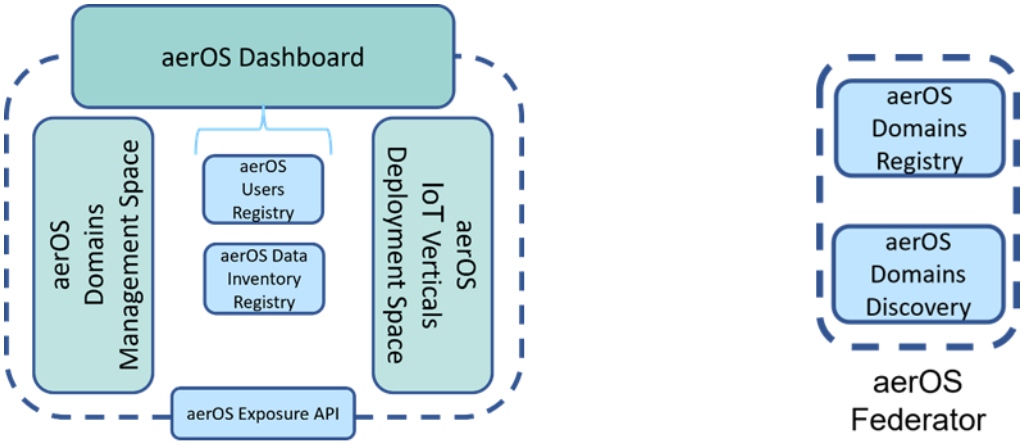


Figure 15. aerOS Management Framework (left: aerOS Management Portal, right: aerOS Federator within aerOS domains)

As a summary, regarding the aerOS architecture advancement, it is advancing at a very healthy speed, and has several conclusions for implications toward the integration (goal of this document).

- Main concerns of aerOS as meta-OS have been identified. The establishment of a continuum, by employing and integrating a compute and network fabric and a service fabric which are vertically providing data to and supported from the data fabric, from edge to cloud incorporating IoT devices is well designed based on rock steady architectural concepts and innovative technologies.
- There is a clear vision and design regarding user demands and interactions with the system. Taking users as starting point aerOS architecture has identified required interactions and has specified the points of access to the ecosystem.
- Decisive enabling concepts having been thoroughly described and have been placed across aerOS building components. Federation is well designed, and its implementation is based on identified enabling technological components which ensure information distribution in a seamless and interoperable way across all the aerOS domains. Orchestration is designed following and innovative approach for a two-level implementation abstracting thus decision making from decision enforcement and having the capability of orchestrating services across all the federated resources continuum.
- The need for a common model, within aerOS, to exchange data among producers and consumers has been decided and is in the first stage of implementation and will be part of knowledge graph employed by aerOS data fabric to represent and expose underlying data in a way that relates concepts and entities to other concepts and entities, providing thus a fully interconnected federated representation of resources.
- In accordance with the above, for the industry verticals data exposure smart models are also investigated.
- Building blocks have been thoroughly defined and described. IE as the minimum execution unit providing a common runtime that can host aerOS facilities and deployment requests. aerOS domain as a set of IEs which host a common set of basic services that provide all the functionality for connecting the aerOS continuum and be part of the federation and orchestration process.
- The need for common APIs internally and for external access has been discussed and initial designs for a minimal API for aerOS domains access and information exchange are done and internal communication is also designed based on both events mesh (message brokers) and service mesh techniques.

- All the components needed to specify and implement aerOS runtime on top of container capable environments are identified and all aerOS basic services, that need to be within each domain are thoroughly described.
- The way to proceed from architectural design and blueprints to implementation is also identified and based on incremental component deployments, first MVPs (Minimum Viable Products) are deployed, and pilot infrastructure integrations are starting.

More of all what ensures architectural progress, and has supported it until now, is the establishment and regular scheduled, virtual and physical meetings, of a strong technical team and per case along with pilot responsible people to support the integration process of architecture to real environments.

4. Integration tools and integration environments

This section gathers the first, initial, set of proposals in aerOS related to integration standards and practices and also referred to the tools in the whole integration process of aerOS software components.

4.1. Integration standard and practices

4.1.1. Trunk-based development

Trunk Based development (TBD) is a software development approach that emphasises the use of a single branch, typically referred to as the "trunk" or "mainline," as the primary development branch throughout the entire development life cycle. Developers collaborate and commit their changes directly to the main branch, rather than working on long-lived feature branches or topic branches, reducing the overhead of branch management. Also, frequent code integration (multiple times per day) promotes early detection and resolution of integration issues, ensuring a more stable code base. Developers working directly on the main branch foster better collaboration, as they have greater visibility into each other's changes and can address conflicts or issues promptly.

The Trunk Based development approach advocates for using feature toggles or feature flags to selectively enable or disable functionality in the trunk. This allows features to be developed and integrated incrementally without impacting the stability of the main branch.

This development practice aligns well with continuous delivery practices, as it emphasises a code base that is always in a releasable state. The frequent integration and small batch sizes enable faster and more reliable delivery of software updates, allowing issues to be identified and resolved early in the development process.

4.1.2. Automated testing

Automated testing plays a crucial role in ensuring the reliability, efficiency, and accuracy of software development processes. There are various types of automated tests that can be performed during software development: unit tests, which verify the functionality of individual units of code; integration tests, which validate the interaction between different components or modules; functional tests, which verify the behaviour of the software from end-to-end.

The code base needs to have a good test coverage level. Unit and integration tests are demanded; functional tests will be managed as needed.

On a further level, contract testing will be used to ensure the API's compliance with the expected contract or specification. It involves verifying that the API code conforms to the defined contract, such as OpenAPI (formerly known as Swagger) specifications. Contract testing tools can be utilized to define and validate contracts between API consumers and providers.

Automated tests will be executed as part of the CI/CD pipeline to ensure that software changes do not introduce regressions or defects. In aerOS, such testing will be aligned and will comply with the DevPrivSecOps methodology that is being defined in task T2.4.

It is worth noticing that automated testing cannot fulfil QA requirements. The main goal should be to have a balanced approach that combines automated testing with manual testing for comprehensive software quality assurance.

4.1.3. Telemetry-first approach

In order to develop and enhance a distributed system is crucial to anticipate problems. This is especially true for the aerOS project considering its nature of meta Operating System. To achieve this, telemetry data and strategies must be defined and took place from the beginning.

The "telemetry-first" approach in software development refers to a methodology that emphasises the collection and analysis of telemetry data from the earliest stages of development. Telemetry data includes various metrics and measurements captured from software applications or systems during their operation. This approach aims to drive decision-making, improve software quality, and enhance user experience based on real-time insights derived from telemetry data.

To be effective in data collection it is crucial to identify and define relevant Key Performance Indicators (KPIs) based on the objectives and requirements of the specific aerOS component or module. KPIs can include response times, error rates, user engagement metrics, resource utilization, or any other measurements that provide insights into the application's performance and user experience. The Integration team in aerOS will make sure that such KPIs (to be identified during the next months) will be aligned with the globally defined technical KPIs of the components of aerOS architecture (expressed in the Grant Agreement).

With the help of monitoring tools, dashboards, and data analysis techniques it will be possible to gain insights into the software's behaviour, performance bottlenecks, usage patterns, and potential issues. This enables proactive detection and resolution of problems.

Also, specific KPIs and telemetry data will be defined and collected for the AI based tasks performed inside the aerOS services.

4.1.4. Configuration is code

Configuration as Code (CaC) practice involves managing and provisioning system configurations using code, in the form of configuration files or scripts. Configurations as code artifacts can be versioned, reviewed, and deployed alongside application code. It brings the benefits of code management practices, such as version control, automation, and collaboration, to the realm of system configuration management.

Configuration code should be integrated into automated deployment pipelines and treated as an integral part of the software release process. This ensures consistency and reproducibility in configuration deployment, reducing the risk of manual errors and simplifying the deployment process. Configuration as Code aligns well with the Infrastructure as Code (IaC) approach, where infrastructure provisioning and configuration are managed using code. Both practices enable consistent and repeatable infrastructure deployments.

Some tools and technologies that support the Configuration as Code practice are Ansible (for configuration management) and tools like Puppet and Chef (for system configuration automation). Further research on these tools and their potential usage in aerOS will take place in the following months within T5.1.

4.1.5. Communications strategy (interoperability strategy)

API contracts serve as a communication and coordination tool between API providers and consumers. They establish a common understanding and agreement on how the API should be used, ensuring interoperability, consistency, and predictable behaviour across aerOs systems and clients (both internal and external).

In aerOS, a whole task (T3.2) is dealing with the design of communication and service API mechanisms, striving for defining the proper methodology and tools to specify the available API endpoints, their URLs, and the supported HTTP methods (GET, POST, PUT, DELETE), or others, together with the operations that can be performed on each endpoint. They should also include data models or schemas that describe the structure, properties, and constraints of the data exchanged between the API and its consumers. This will help ensure consistent data representation and enables validation of request and response payloads.

From T5.1, several suggestions will come forward, so that communications in APIs can be formalised. API contracts could be documented using specification formats like OpenAPI (previously known as Swagger), RAML (RESTful API Modelling Language), or API Blueprint. These formats provide a standardized way to document and describe APIs, making it easier to generate client SDKs, perform automated testing, and provide interactive API documentation.

4.2. Integration tools

This section will cover the tools that will be used in the project in order to carry out the CI/CD. In aerOS and specifically in the deliverable D2.4, the first version of the DevPrivSecOps methodology that will be used for the development of the software and its operation in the project has been defined.

4.2.1. Continuous integration and development tools

This methodology will allow the project's developers to provide the appropriate practices in the life cycle of the software developed during the project. Especially, and taking into account that the DevOps methodology is known and applied by most developers, in the first version of the methodology delivered in D2.4, emphasis has been put on including tests to analyse security and privacy. With these tests, it is possible to analyse the security and privacy of the software in the different phases of the DevOps methodology, identifying problems at an early stage, with the intention of mitigating them as soon as possible. The use of this methodology will allow aerOS to create secure and privacy aware software by design.

One of the critical components of the DevPrivSecOps methodology is Continuous Integration (CI) and Continuous Delivery (CD): CI/CD are used to automate the delivery process from code development to deployment of the software component. Continuous integration analyses through different tests (including security and privacy related tests) the automatically generated code and integrates it with the other software components already analysed, while continuous delivery automates the entire software release process up to the deployment of the software in the production environment. The benefits of CI/CD are accelerated delivery, faster problem resolution, reduced risk of bugs and vulnerabilities, and improved code quality. In addition, DevPrivSecOps addresses vulnerabilities in software development by inserting security and privacy audits and penetration testing at every step of the process.

As described in D2.4, GitLab has been the selected tool for version control and CPD in aerOS. In GitLab, the CI/CD process is called CI/CD pipelines, composed of jobs (defining the action, e.g., compile or test code) and stages (grouping a series of jobs and defining the exact time to execute them, e.g., stages containing test jobs are executed after the stage that compiles the code). Jobs are the most essential elements of a GitLab pipeline as they are the ones that perform the required executions. Jobs are not limited by their number within a stage and can be defined with restrictions specifying the conditions of their execution. Pipelines typically progress to the next stage if all jobs in a stage are successful; otherwise, the next stage is not executed, and the pipeline terminates before the completion of all stages. According to GitLab's official documentation, a typical pipeline consists of four stages: build, test, stating and production. Moreover, pipelines can be triggered based on a wide range of conditions: scheduled in time, manually triggered (by HTTP requests, jobs from other pipelines, webhooks, ...) or commit changes in a branch, among others.

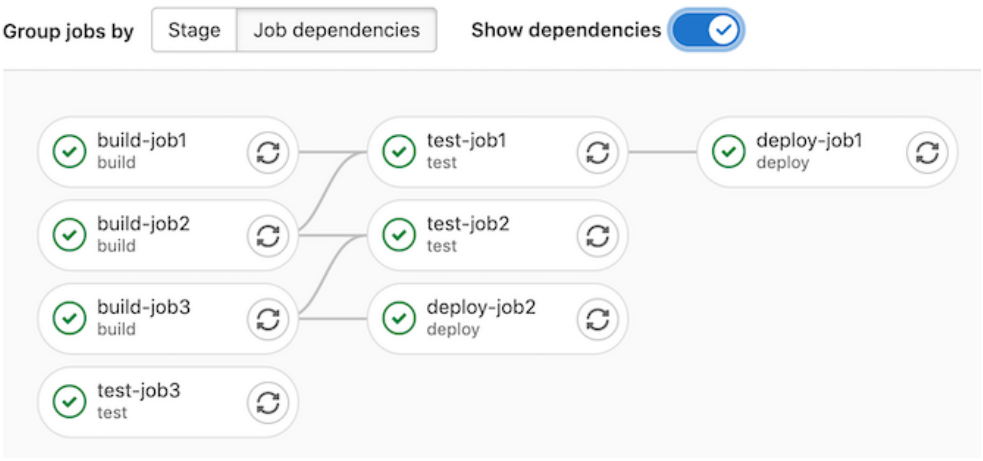


Figure 16. Gitlab CI/CD pipeline composed by jobs and stages

In Gitlab (Figure 16. Gitlab CI/CD pipeline composed by jobs and stages**),** the CI/CD process is carried out by an executor, which runs a series of jobs listed in a YAML file, the *.gitlab-ci.yaml* file, and reports its final results to a dashboard to see the status of the pipeline executions in real time.

For the CD, GitLab has special runners to be able to deploy the code that has been developed in a test environment first and in production environment at the end of the DevOps process. In the case of aerOS, a runner (or more than one) will be installed in a cloud environment provided by the partner CloudFerro allowing to automate and test the generated code (as it can be seen in the Figure 2. Adaptive iteration steps). Such cloud development and testing environment provided by CloudFerro is currently being designed and set up and will be documented properly in next iterations of this deliverable. This procedure is called GitOps and GitLab recommends using the FluxCD[1] tool that is integrated in its pipeline to perform this task. FluxCD (Figure 17. GitLab + FluxCD) is a tool to keep Kubernetes clusters in sync with configuration sources (such as Git repositories) and automate configuration updates when there is new code to deploy.

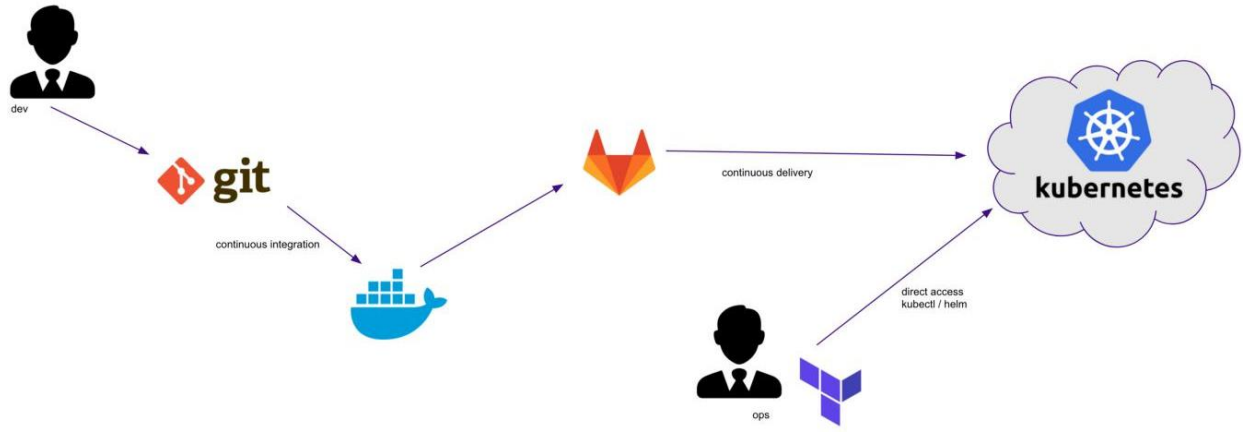


Figure 17. GitLab + FluxCD

4.2.2. Potential integration tools analysis

There are numerous integration tools available in the market, each with its own unique features and capabilities. Based on the integration standard and practices specifications, here it is reported a potential link of integration tools that can be used in aerOS integration flow. The judicious selection of those will take place during the next few months, and will be properly reported in the next deliverable of the WP (D5.3 and D5.2).

[1] https://fluxcd.io/

4.2.2.1. SonarQube

SonarQube is an open-source platform for continuous code quality inspection and static code analysis. It provides a comprehensive set of tools and features to analyse code bases, detect bugs, vulnerabilities, and code smells, and measure code quality against defined standards and best practices. It could be very useful to help guarantee a high quality of code and measure test coverage in aerOS, but its usage will be constrained to the alignment with the decisions to be taken for the DevPrivSecOps methodology.

It supports a wide range of programming languages, including popular languages such as Java, C/C++, C#, JavaScript, Python, Ruby, TypeScript, and many others. It provides language-specific analyzers and rules to cater to the specific characteristics and best practices of each language.

Also, it integrates with CI/CD pipelines, allowing automatic code analysis and reporting as part of the development process. It can be configured to perform code analysis on every code commit or as part of scheduled builds, ensuring that code quality is monitored continuously.

It is also worth considering that SonarQube offers customisation options to adapt to specific project requirements. It allows the creation of custom rules, extensions, and plugins to integrate additional analysis tools, security scanners, or specific code quality rules.

4.2.2.2. OWASP Zap

OWASP Zap (Open Web Application Security Project Zed Attack Proxy) is an open-source security testing tool designed to help identify and mitigate security vulnerabilities in web applications. It is a widely used penetration testing tool for web application security assessments. It can perform both active scanning (sending malicious requests to identify vulnerabilities) and passive scanning (observing application responses for potential issues). It can intercept and modify requests and responses in real-time for vulnerability identification and testing.

OWASP Zap is highly extensible and offers an API and various add-ons that can be used to customize and enhance its functionality. It integrates well with other security testing tools, development environments, and CI/CD pipelines.

4.2.2.3. Sentry

Sentry is an open-source error tracking and monitoring platform that helps developers identify, track, and debug issues in their applications. It provides real-time error monitoring and reporting, allowing developers to proactively detect and resolve issues before they impact users. Sentry supports various programming languages and platforms, making it versatile and widely used in the software development community.

Sentry captures and records error events that occur within an application. It collects information such as error messages, stack traces, request data, user context, and other relevant metadata. These details help developers understand the cause and context of errors, enabling effective debugging and resolution. The tool seamlessly integrates with popular development tools and workflows, such as source code repositories, issue trackers, and collaboration platforms. This allows for streamlined error management and efficient collaboration among development teams.

4.2.2.4. Prometheus

Prometheus is an open-source monitoring and alerting software designed to monitor the health, performance, and availability of systems and applications in a highly scalable and flexible manner. Prometheus follows a pull-based model, where it scrapes metrics from targets (such as servers, containers, or services) and stores the collected data in a time-series database.

Prometheus provides a powerful query language called PromQL (Prometheus Query Language), which allows users to retrieve and aggregate metrics based on specific criteria. PromQL supports functions, operators, and selectors for querying and manipulating time-series data.

The tool integrates well with GitLab and other monitoring and alerting systems. Also, it can be combined with other tools, such as Grafana, to create visualizations and dashboards for monitoring and analysis. Grafana can query Prometheus and present the data in customizable and interactive dashboards.

4.2.2.5. Grafana

Grafana is an open-source data visualization and monitoring platform that allows users to create and display interactive, real-time dashboards for various data sources. It supports the exploration, analysis,

and presentation of data from different systems and databases, making it a popular tool for monitoring and observability.

The tool provides a flexible and intuitive user interface for creating dashboards. Users can drag and drop various panels (graphs, tables, gauges, etc.) onto the dashboard canvas and configure them to display specific data. Grafana supports a wide range of visualization options, including line charts, bar charts, pie charts, and heat maps.

Also, Grafana provides a flexible and intuitive user interface for creating dashboards. Users can drag and drop various panels (graphs, tables, gauges, etc.) onto the dashboard canvas and configure them to display specific data.

5. Conclusions

D5.1 has presented a first set of integration and development tools that will be analysed to be used in aerOS to achieve the ultimate goal of integrating each software service, module and component of the aerOS system into a single ecosystem responding effectively to the application's functions, requirements and use cases.

There has been made the proposal to rename the deliverable, "Integration approach and methodology" as it should have been established since the beginning to better represent project's implementation stage at M12 of the action. The integration plan for the whole aerOS system will be depicted in D5.2 Integration, evaluation plan and KPIs definition.

The deliverable addressed two critical aspects of the system integration: the strategic one, illustrating the overall approach and the technical one, specifying the technical tools for a successful integration.

Concerning the strategy, it has been chosen an adaptive methodology instead of a predictive approach, given the complexity of the project and level of uncertainty. It has been decided to create an initial backlog of services and functionalities that will be validated by the entire consortium and prioritized with the help of the necessary stakeholders. Recurrent meetings to define the work in the different iterations will be organised. Roles and responsibilities for this process have been identified and appointed, bearing in mind the profiles of the users in the GitLab repository of aerOS (serving as the collaborative repository in the project). In order to facilitate the collaboration, organisation and communication among the different team members, a set of Integration Management Platforms have been selected to support the team in the "project management and issue tracking" and in the "collaboration and documentation" aspects.

Regarding the technical dimension, a first version of the aerOS system integration has been outlined with main components framed with security and privacy mechanisms: the Infrastructure Element and the aerOS domain. The former is the fundamental building block providing the computational infrastructure and the latter is a complete aerOS domain hosting and sharing among all its IEs aerOS basic services. Among these basic services are the data fabric provisioning, orchestration and federation services as described before, security services, networking facilities.

Finally, a first list of integration standards and tools were identified and presented in the deliverable. This list will be further discussed with the consortium partners in the next project phase to select the most suitable solutions for the aerOS system.

To sum-up, as already highlighted, the integration plan for the technical outcomes of aerOS (out of WP3 and WP4) will be implemented in the next phase, through the DevPrivSecOps approach, which emphasises the individual responsibility of individual development teams to succeed. This means that all technical partners have a significant share of responsibility for the overall performance of the aerOS system, quality of development, and timely releases for the next Milestones, specifically regarding the deliverable 5.2 - Integration, evaluation plan and KPIs definition (2) foreseen at M24.

Also, the next months in T5.1 will cover some aspects that have remained to be addressed (due to logical organisation of the task execution). Those will be, among others, how to package the results (or how this packaging aspect will be dealt with), how to document the results (or how the documentation will take place - e.g., with ReadTheDocs), a further detailed list of best practices to be followed (e.g., Linux CII Best Practices) and the emphasis on the continuous development/integration environment to provided by CloudFerro.

In addition, in order to successfully complete the entire integration process and store any progress securely, the technical partners will coordinate and perform part of the development, testing, and integration through specific tools. To use these tools or to gain access to their services, a guidance document will be provided explaining how to access and use specific accounts or licenses.