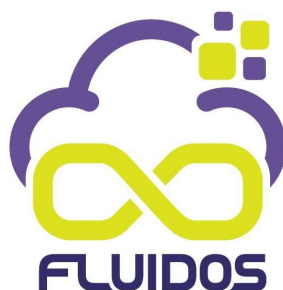


Grant Agreement No.: 101070473
 Call: HORIZON-CL4-2021-DATA-01
 Topic: HORIZON-CL4-2021-DATA-01-05
 Type of action: HORIZON-RIA



D9.1 DEVELOPMENT ENVIRONMENT AND CI/CD WORKFLOWS

Revision: v.0.3

Work package	WP 9
Task	Task 9.1
Due date	28/02/2023
Submission date	31/01/2023
Deliverable lead	UMU
Version	0.3
Authors	Antonio Skarmeta (UMU)
Reviewers	Andy Edmonds (TER)



Abstract	Documentation of the CI/CD environment and the tools selected.
Keywords	Agile Integration, testing, continuous integration, continuous delivery

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributor(s)
V0.3	31/01/2023	1st version of the document	Eduardo Cánovas (UMU) José Francisco Pérez (UMU)

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the "Flexible, scaLable and secUre decentralizeD Operating System" (FLUIDOS) project's consortium under EC grant agreement 101070473 and do not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

COPYRIGHT NOTICE

© 2022 - 2025 FLUIDOS Consortium

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	R	
Dissemination Level		
PU	Public, fully open, e.g. web	X
SEN	Sensitive, limited under the conditions of the Grant Agreement	
Classified R-UE/ EU-R	EU RESTRICTED under the Commission Decision No2015/ 444	
Classified C-UE/ EU-C	EU CONFIDENTIAL under the Commission Decision No2015/ 444	
Classified S-UE/ EU-S	EU SECRET under the Commission Decision No2015/ 444	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc

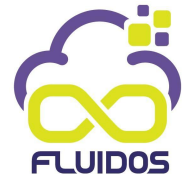
DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.





EXECUTIVE SUMMARY

The purpose of this document is to provide a single point of reference on the project procedures, and acts as an initial report and the deliverable outcome regarding the activities of WP9 “FLUIDOS Agile integration & testing”. Its objective is to accompany and document the technical requirements related to the FLUIDOS CI/CD platform as well as the FLUIDOS Development and Testing Environment as used in the deployment and testing of the FLUIDOS artefacts.





TABLE OF CONTENTS

EXECUTIVE SUMMARY	3	TABLE		CONTENTS
LIST OF FIGURES	4	LIST		TABLES
ABBREVIATIONS	6			71
INTRODUCTION				81.1
Deliverable Overview				81.2
Document Structure				81.3
Document Scope				91.4
General objectives				92
AGILE INTEGRATION				102.1
Sprint Management and Agile Scrum Concepts				102.1.1
Organisation				102.1.2
Sprints				122.1.3
Agile Tools				132.2
Documentation of the possible Software Tools				133
CONTINUOUS INTEGRATION				163.1
DevOps and CI/CD practices				163.2
Overview of Continuous Integration/Continuous Delivery Selected Software Tools and Environment				173.3
Continuous Integration/Continuous Delivery Stack and Methodology				173.4
CI/CD Scenarios and selected tools				173.4.1
Scenario 1 (Self Managed)				193.4.2
Scenario 2 (Lightweight Alternative)				223.4.3
Optional Components				244
SECURITY PRACTICES FOR THE CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY PLATFORM AND DEVELOPMENT & TESTING ENVIRONMENT				265
CONCLUSIONS				28
				REFERENCES
	29			



LIST OF FIGURES

FIGURE 1: BUILD-MEASURE-LEARN WORKFLOW	10	FIGURE 2: SCRUM TEAM	12	FIGURE 3: SCENARIO 1	
FIGURE 4: CI/CD TOOLS USED IN SCENARIO 1 FLUIDOS	20	FIGURE 5:	21		
SCENARIO 2 WORKFLOW	22				

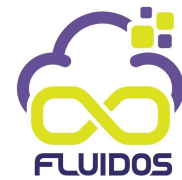




LIST OF TABLES

TABLE 1 : SCENARIO 1 CORE TOOLS 22
24

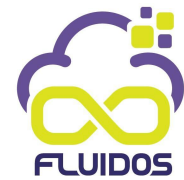
2 : SCENARIO 2 CORE TOOLS



ABBREVIATIONS

CA	Certificate Authority
CD	Continuous Deployment - Continuous Delivery
CI	Continuous Integration
Dx	Deliverable (where x defines the deliverable identification number, e.g., D1.1)
DevOps	Software Development (Dev) and IT Operations (Ops)
DoS	Denial of Service
EU	European Union
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
ICMP	Internet Control Message Protocol
LAND	Local Area Network Denial
MitM	Man-in-the-middle
SCM	Source Code Management
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
SSO	Single sign-on
SYN	Synchronized
Tx	Task (where x defines the deliverable identification number, e.g. T1.1)
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
VCS	Version Control System
WP	Work Package





1 INTRODUCTION

1.1 DELIVERABLE OVERVIEW

This report will act as guidelines and a broad documentation of the FLUIDOS CI/CD platform as well as the Development and Testing environment which will be created and configured in the concept of Task 9 FLUIDOS platform Development Environment Setup.

As per the requirements of task 9.1 it is expected to:

- Guide the collaborative development and integration of FLUIDOS components.
- Setup a Continuous Integration/Continuous Development (CI/CD) environment to ensure higher quality software.
- Integrate all software components developed in technical WPs and release the FLUIDOS platform in two major releases.
- Test FLUIDOS at different levels (unit, system, integration, and end-to-end tests) and monitor its progress toward its KPIs.
- Ensuring the quality of the produced integrated platform.
- Deliver an integrated solution ready for the experimental validation in WP7.

In this report and based on the above requirements, the FLUIDOS CI/CD platform is presented and described. The infrastructure consists of a set of software components that incorporate and align with the modern Agile methodology and DevOps procedures, able to support all necessary development and testing activities.

1.2 DOCUMENT STRUCTURE

To incorporate the above information and guidelines the report is organised in the following subsections/chapters:

- Chapter 1 provides an overview and structure of the document. Chapter 2 describes the Agile Integration and overview of all Software tools of the project.
- Chapter 3 describes the FLUIDOS CI/CD, the tools selected for the software development teams and established Continuous Integration/Continuous Deployment workflows.
- Chapter 4 discusses Security practices for the Continuous Integration/Continuous Delivery platform and Development & Testing Environment.





1.3 DOCUMENT SCOPE

The results presented in this document are in adherence with the best practices, methodologies and tools. Reliable and widely used by the developer community, open-source tools are utilised, following the common set of DevOps methodologies, in order to support holistically the development, testing, integration and deployment processes. Finally, we ascertain to customise and deploy the corresponding open-source tools to fulfil the needs and requirements that uniquely characterise the FLUIDOS's dedicated environment.

1.4 GENERAL OBJECTIVES

FLUIDOS is a research-oriented project, whose results must be confirmed and benchmarked with real-world integrated prototypes. From one side, FLUIDOS foresees the necessity to define strict coding, testing and integration rules to guarantee the delivery of high-quality artefacts (albeit still at a prototype level, $TRL \leq 5$); on the other side, most of the personnel involved in FLUIDOS are not full-time developers, hence tools with reasonable learning curve should be privileged over highly powerful, but complex CI/CD platforms.

Consequently, the project will choose the required tools based on the following principles.

1. **Keep small the number of tools.** The more tools we use, the longer the learning curve, and the hardest becomes to connect and correlate information present in one tool with the one in another (e.g., code in GitLab, bug tracking in Kira, testing results in Jenkins).
2. **Privilege well-known tools** (or the one that, given the desired levels of features, offer a more favourable learning curve) against more specialised solutions, perhaps more appropriate for production-grade CI/CD pipelines.
3. **Privilege managed services.** Keeping the CI/CD infrastructure up-to-date, as well as guaranteeing the proper security patches, run-time features, etc., is a time-consuming task that has nothing to do with research. Therefore, whenever possible, FLUIDOS would select services that are managed by third-parties, which can be simply consumed (i.e., used) by the project researched and possibly by external contributors (depending on the cost).
4. **Facilitate the testing of the software artefact within the several physical domains in which FLUIDOS operates.** We foresee FLUIDOS artefacts to be deployed on robots, on embedded devices, on managed cloud providers. Ideally, the chosen testing solution should be able to support them all.

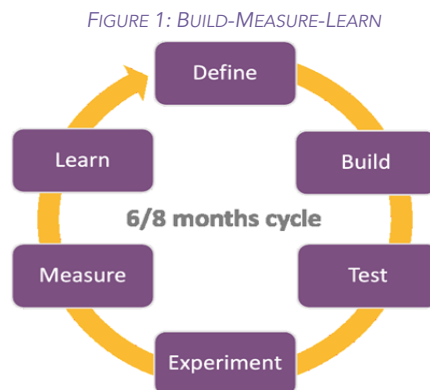


2 AGILE INTEGRATION

2.1 SPRINT MANAGEMENT AND AGILE SCRUM CONCEPTS

FLUIDOS uses an agile methodology to drive the research and innovation activities, designed to transform innovative ideas into profitable products according to real user/customer needs. To achieve this goal, the methodology focuses on learning and discovering how to fit a technology into the market instead of how to carry out the technological developments themselves.

It is based on executing iterations on a Build-Measure-Learn (formalised in the Lean methodology) feedback loop, which provides a scientific method for validating two aspects: that the implemented technology is valid (i.e., the implementation is right); that it responds to real needs (i.e., it is the right implementation). This drives the decision process by objective data and it also pushes developments in the direction of what potential users and customers perceive as valuable.

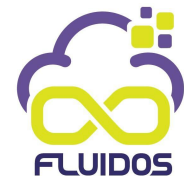


2.1.1 Organisation

Due to the nature of this project the most convenient organisation is component-based (one component, one team). This organisation is aligned to the co-location of the teams, it will simplify the communications, reducing the handoff of requirements, designs, and test data.

Each team will be able to aggregate the needs of multiple features into the architecture of their component and can focus on building the best-possible component. The team will manage its own backlog.

Following Agile there will be defined the following roles:



Product Owner:

- Define the overall strategy.
- Prioritises work from the product backlog list, aligned with the overall strategy.
- Defines goals, answering any questions.
- Oversees the evolution of the components.
- Understands the requirements.
- Evaluates the progress through each iteration.

In FLUIDOS the owner of each component will be its Product Owner.

Scrum Master:

- Ensures the team adheres to the Agile practises the team has agreed to follow.
- Gives support to the Product Owner
- Ensures that everyone on the team understands goals.
- Finds techniques for effective product backlog management.
- Helps the team to see the necessity for clear and concise product backlog items.
- Understanding product planning
- Ensuring the Product Owner knows how to organise the product backlog.
- Understanding and practising agility
- Facilitating events as requested or needed.
- Implements the Scrum approach with development teams.

The Product Owner of each component will assign a Scrum Master to the team.

Development and Tester Team:

Using the backlog items of the sprint the development team will:

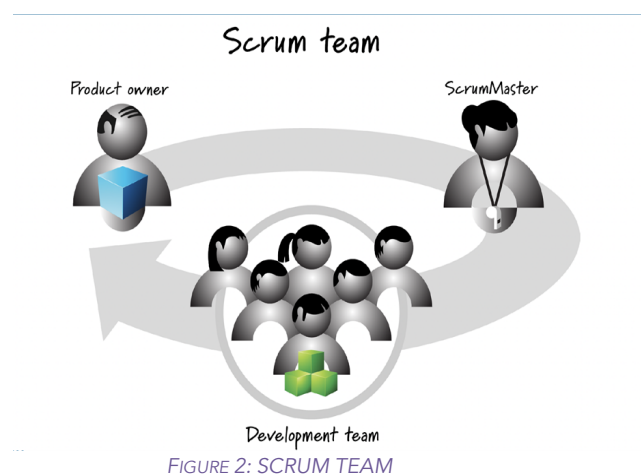
- **Define** the tasks for the backlog items.
- **Realise** the tasks.
- Create and test the code.

An additional team will be needed, the **Integration Team** created from WP9. This team will:



- Set up a continuous integration environment and provide guidance to the component owners for the continuous integration and testing of their components.
- Test all the features at the system level.
- Test non-functional and quality requirements. This includes performance and reliability testing.

The Scrum Master of the Integration Team will be a **Global Scrum Master**, this role will facilitate the coordination between components.



2.1.2 Sprints

The activities will be organised in **sprints** lasting 1 month that will allow results to be quickly assessed and any appropriate adaptations made. At the end of each Sprint the achievements will be presented during the Sprint Review. From these sprints we will get updates and achievements, issues, and next steps.

For each FLUIDOS team at the beginning of each sprint:

- The product owner will set the priorities from the backlog.
- Identify the task list for the sprint.
- Elaborate the tasks schedule.

Biweekly reviews

Biweekly, the status and progress of components will be monitored reviewing the task list status (completed tasks, new task identification) and the task progress.

To these reviews will assist:

- Scrum Master
- Development/Tester Team

Reports of these reviews will be sent to:



- Scrum Master
- Development/Tester Team
- Product Owner

2.1.3 Agile Tools

Agile Tools for fast communication

Fast communication tools will be used for each module or component, for example one **Slack channel** for each module or for each component.

Agile Tools for work management

The greater the levels of transparency within the team, the easier it is for the Scrum Master to monitor progress and minimise misunderstandings. If it looks like team members aren't communicating effectively or work is falling behind, the Scrum Master can step in and help resolve any differences.

Recommended software tools for work management are **Jira**, **Zenhub** or **Taiga**

2.2 DOCUMENTATION OF THE POSSIBLE SOFTWARE TOOLS

The possible software tools that could comprise The FLUIDOS platform are listed below, namely:

- **GitHub** (GitHub Inc., n.d.) is an internet-hosting service for software development and version control, a software tool that helps software teams manage changes to source code. It offers distributed version control of Git as well as access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project (GitHub Inc., n.d.). Version control, also known as source control, is a practice commonly used by developers' teams in order to track and manage changes to software code. As development environments have accelerated, version control systems help software teams work in an agile manner which is especially useful for DevOps teams (Atlassian, 2021). It can help to reduce development time and increase successful deployments with less disruptions and errors.
- **Gitlab** (GitLab Inc.) GitLab is an open source code repository and collaborative software development platform for large DevOps and DevSecOps projects. GitLab is free for individuals. GitLab offers a location for online code storage and capabilities for issue tracking and CI/CD. The repository enables hosting different development chains and





versions, and allows users to inspect previous code and roll back to it in the event of unforeseen problems.

- **Jenkins** (Jenkins, n.d.) an open-source tool automation server that supports building, deploying and automating of the project acting as the “brain” of the process. Jenkins has been configured to respond to developer related actions and triggers and initiates software build, testing and packaging as well as “pushing” images to a dedicated registry.
- **Travis** (©Travis Ci, gmbh) Travis CI is a hosted continuous integration and deployment system.
- **Jira** (Atlassian Corp.) is a software application used for issue tracking and project management. The tool, developed by the Australian software company Atlassian, has become widely used by agile development teams to track bugs, stories, epics, and other tasks.
- **Zenhub** (Axiom Zen) is an agile project management and product roadmaps tool designed to help software teams organise, plan, track, and manage their work. ZenHub is available either inside GitHub via a browser extension or as a standalone web app.
- **Taiga** (Taiga Agile, LLC) Taiga is an open-source project management software for multidisciplinary teams that work agile across both scrum and Kanban frameworks.
- **RobotFramework** (Robot Framework Foundation) Robot Framework is a generic open source automation framework. It can be used for test automation and robotic process automation (RPA). Robot Framework is supported by Robot Framework Foundation. Many industry-leading companies use the tool in their software development. Robot Framework is open and extensible. Robot Framework can be integrated with virtually any other tool to create powerful and flexible automation solutions. Robot Framework is free to use without licensing costs.
- **SonarQube** (SonarSource) is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs and code smells on 29 programming languages. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security recommendations. SonarQube can record metrics history and provides evolution graphs. SonarQube provides fully automated analysis and integration with Maven, Ant, Gradle, MSBuild and continuous integration tools (Atlassian Bamboo, Jenkins, Hudson, etc.)
- **JFrog Container Registry** (JFrog, n.d.) a repository manager tool used to organise resources, it will provide a private registry (Docker) in which The FLUIDOS project's artefacts will be securely stored and distributed.
- **Keycloak** (Keycloak-Team, n.d.) an open-source Identity and Access Management tool which is used to add authentication to applications and secure services. All other components of the FLUIDOS CI/CD platform can use the Keycloak authentication to gain





access to their services. This way the users can authenticate with Keycloak rather than individual applications in a centralised manner.

- **SonarQube** (SonarQube, n.d.) an open-source platform that provides automated, continuous inspection of code quality detecting bugs as well as security vulnerabilities and supports various programming languages.
- **NGINX** (NGINX, 2022b) an open-source web server designed as a reverse proxy. For the FLUIDOS CI/CD services. A reverse proxy can retrieve resources from servers on behalf of a client, thus protecting the FLUIDOS CI/CD platform services.
- **Portainer** (Portainer.io, n.d.) is an open-source management, UI tool for Docker hosts. It allows for management of different Docker environments, offering the host's administrator various monitoring and management options for all the Docker resources (containers, images, volumes, networking, etc.). It is a single container, which offers a web interface running on any Docker engine, either standalone or in swarm mode.
- **Slack** is an instant messaging program designed by Slack Technologies and owned by Salesforce. Although Slack was developed for professional and organisational communications, it has been adopted as a community platform. Users can communicate with voice calls, video calls, text messaging, media and files in private chats or as part of communities called "workspaces". In the FLUIDOS platform can be used as a tool to send alerts both in the execution of Jenkins jobs as well as alerts to monitor crashes in any of the deployed pods.
- **Jira** (Atlassian Corp.) is a software application used for issue tracking and project management. The tool, developed by the Australian software company Atlassian, has become widely used by agile development teams to track bugs, stories, epics, and other tasks.
- **Zenhub** (Axiom Zen) is an agile project management and product roadmaps tool designed to help software teams organise, plan, track, and manage their work. ZenHub is available either inside GitHub via a browser extension or as a standalone web app.
- **Taiga** (Taiga Agile, LLC) Taiga is an open-source project management software for multidisciplinary teams that work agile across both scrum and Kanban frameworks.





3 CONTINUOUS INTEGRATION

3.1 DEVOPS AND CI/CD PRACTICES

FLUIDOS needs CI/CD as it is the optimal practice in the development process regarding the tools used and composing the FLUIDOS platform. Continuous integration (CI) as the term suggests allows for frequent integration and versioning of code as well as continuous checks and code verifications (Zhao et al., 2017). This will allow for smooth integrations of components that will be successfully incorporated and most importantly verified and tested ensuring this way that new and functional commits will be merged in an already error-free environment/application. CI succeeds in minimising the release cycle and allows the developers to discover and fix bugs early, therefore avoiding backtracking and code validation while providing them with more development and integration time (Zhao et al., 2017).

Continuous Delivery is strongly connected with CI acting as the next step to the CI/CD pipeline. In this stage it is ensured that the application is ready for delivery to the end-users providing “packaging” of the application. Moreover, during this stage tools that are responsible for automated building and releasing of the application are ensured, keeping this way the artefacts always ready for deployment at any given time. Continuous Delivery ensures that the application can be released correctly and more frequently, regardless of code changes which for that fact are kept small-scaled.

Ultimately, Continuous Deployment, the final stage of the CI/CD procedure, provides automated launching and distribution of the applications (and its components). For every change that has passed the previous stages and therefore is able to be incorporated to the end-result, a new deliverable/output/version is released, while in case of a failed test through the pipeline the change is prevented from deployment. This process is fully automated and as such does not require any human intervention.

Overall, the stages as described above (CI, CD & Continuous Deployment) are the optimal practices regarding Development & Operations. This process focuses heavily in establishing improved and constant collaboration and communication between development and operation teams thus achieving the required quality and business objectives. The discussed processes and environment facilitate the automations of large components related to development, testing and deployment.

The FLUIDOS developer teams will utilise these tools and procedures to commit changes, build and test source code and project artefacts in the most efficient manner. The Continuous Deployment stage in FLUIDOS will be executed in testing servers (Development & Testing Environment). No production servers during the writing of this deliverable.





3.2 OVERVIEW OF CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY SELECTED SOFTWARE TOOLS AND ENVIRONMENT

Interconnected software components make up the automated build system for FLUIDOS, and mutually perform a series of operations to provide correct integration and validation of newly introduced parts or elements by the developer teams. Changes made by developers on individual tools of the FLUIDOS project, trigger the CI/CD platform, which subsequently integrates them in the FLUIDOS platform.

The deployment of the individual software components that constitute the FLUIDOS Continuous Integration/Continuous Deployment (CI/CD) platform will be deployed on the cloud. All server's network traffic (inbound & outbound) will be secured by the proper firewall configuration.

3.3 CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY STACK AND METHODOLOGY

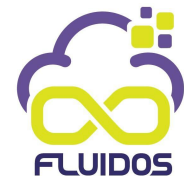
This chapter illustrates and further explains the FLUIDOS CI/CD platform, an environment that incorporates various software components to integrate, test and finally deploy the complete FLUIDOS platform. To facilitate development workflows, the major goal of this activity is to set up, install, and configure a CI/CD infrastructure. To aid with the integration process, a set of cooperative workflow tools is established. Each individual software component is further presented in the following subsections. Furthermore, each specific module is accompanied by guidelines regarding its deployment.

The CI/CD development process can be described as a methodology to frequently deliver software by introducing automation into the stages of development.

3.4 CI/CD SCENARIOS AND SELECTED TOOLS

This section describes all the components and software tools selected in the above-mentioned methodology. The environment is made up of numerous software elements that work together to integrate, test, and eventually deploy the FLUIDOS platform. To illustrate the task each component undertakes towards the end-result; the following subsections provide additional information on each of the necessary software modules comprising the FLUIDOS CI/CD platform.





To satisfy the needs, two scenarios have been proposed, the first that will have to be installed and managed, and the second, on the contrary, that will lack installation and maintenance and can be used more immediately. This will allow us to have a more elaborated and robust version of FLUIDOS against another faster and lightweight version.

For each scenario we will need four Core components:

1. Version Control System (VCS)

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems (VCS) are software tools that help software teams manage changes to source code over time. VCS keeps track of every modification to the code in a special kind of database. It is a remote repository of files that comprise the source code of a software application. If a mistake is made, developers can compare earlier versions of the code to help fix the mistake while minimising disruption to all team members.

2. Continuous Integration/Continuous Delivery (CI/CD)

Continuous integration (CI) it's a primary DevOps practice, it allows for automation of the integration of code changes, from multiple contributors to a single software project. Moreover, it allows developers to regularly commit code into a centralised repository where builds and tests then run thus asserting the new code's correctness before integration (Atlasian, n.d.).

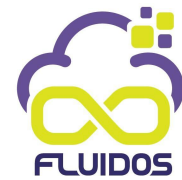
- Developers get copies (locally) of the source code and apply changes to their local system.
- The changes then are committed to the centralised, common repository.
- The server is immediately notified upon any incoming change.
- The server initiates the below actions:
 1. Pulls the latest code version which includes the newly added changes.
 2. Builds the application and reports any potential problems.
 3. Runs unit and integration tests, reporting any issues if they exist.
 4. Releases any artefacts to be deployed for testing.
 5. Assigns a build tag to the software version that was built.

Additionally, due to the above-described process developers benefit by getting aware of issues in a short timeframe and can act on solving the issues after committing a change, this process (CI) is available through the whole duration of the project's lifecycle.

3. Test Logs and Report

A mechanism to display results of unit and integration tests performed after testing builds of the software under development.





4. Container Registry

A stateless, highly scalable central space for storing and distributing container images. They provide secure image management and a fast way to pull and push images with the right permissions.

3.4.1 Scenario 1 (Self Managed)

In this section, a chain of tools is proposed that will compose a scenario that is elaborated to install and manage, but can provide more flexibility and functionality in the future. On the other hand, we will need resources to carry out the installation of all the components that make it up.

3.4.1.1 Tools Selected

- Version Control System: GitLab

GitLab will be used as the VCS within the FLUIDOS CI/CD platform. It is an easy yet powerful and intuitive git VCS. Multiple developers can concurrently create, merge, and delete parts of the code as they are working independently at their local system, before applying the changes to the shared GitLab repository. GitLab offers a set of useful features to the software development process, such as version control, issue tracking, code review, wiki, etc.

Like every other git VCS, GitLab comes with extended branching capabilities. In most cases, there is one main branch in a repository from which each developer who works on a specific feature or bug fix creates an additional, diverging branch. Once the developers have concluded their changes on the source code, they subsequently merge their side branch back into the main branch.

Gitlab can be used in two ways, one in which the service is already hosted by the Gitlab company (SaaS, <https://about.gitlab.com/>) and the other in which a self-managed server instance is installed. The second option will be chosen for this scenario to provide more versatility and freedom.

- Continuous Integration: Jenkins

Jenkins uses pipelines to generate an ordered series of events/tasks/actions. These tasks relate to the purpose of building, testing, packaging, deploying, and storing software.

As described previously, *repositories* under the FLUIDOS GIT will be connected to corresponding Jenkins Pipeline jobs. Source code changes in GIT repositories, such as commits, merges etc., will trigger the respective repository pipeline. The pipeline, in turn, defines compilation, build and test stages.

A *Jenkinsfile* is a text file containing the definition (implementation and process) of a Jenkins pipeline. The Jenkins *Pipeline* implements a basic three-stage continuous delivery and consists of a precisely ordered number of steps regarding building, testing and delivering of



an application. The Jenkins server can pull the *Jenkinsfile* current version upon every new build trigger. The FLUIDOS pipelines are defined using declarative syntax format.

- Logs and Report: RobotFramework

The combination of Jenkins with RobotFramework is immediate, since we can find in Jenkins a plugin that prepares the environment to work with RobotFramework (RobotFramework Plugin), the objective is integrating the robot plugin in the testing stage. The main function is to show the outputs of the test executions, a shell command has to be included in a step of the pipeline of the execution script to execute a Robot test.

- Container Registry: Docker Hub (Own Registry)

In this scenario Docker Hub would be used but installing its own registry.

Here are some essential reasons why use own private registry instead of a public registry like DockerHub(SaaS).

- Control where the images are stored - A private registry gives full control over the storage location of the images and how can access them.
- Custom image pipeline
- More privacy for proprietary and private images
- Custom configuration options e.g. logging, authentication, load balancing, etc..

- Build environment: Kubernetes

Kubernetes will help control the deployment of the applications that are being developed to build a functional scenario ready for make executions, unit testing, integration testing in which any state of the application can be built.

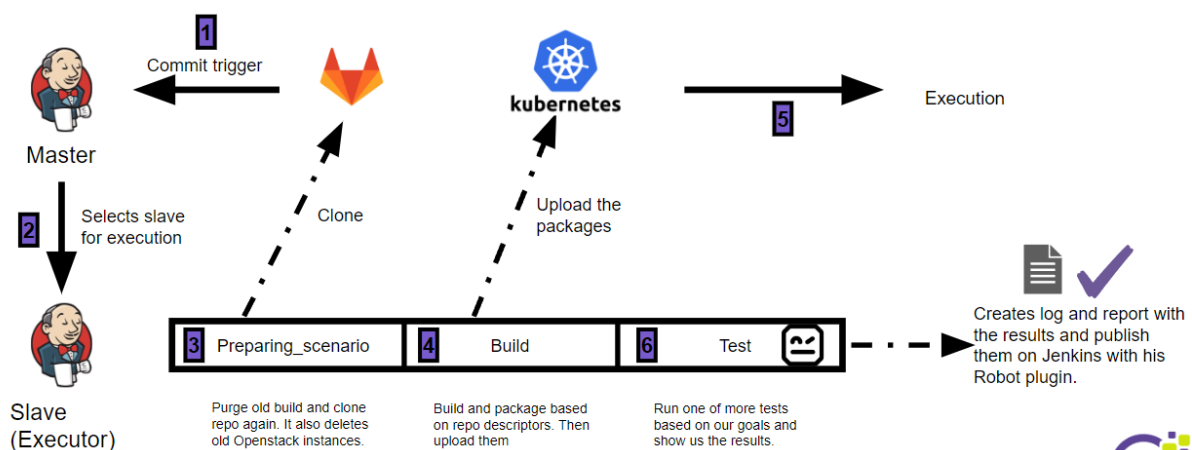


FIGURE 3: SCENARIO 1 WORKFLOW

1. The developer commit code to the project code repository in GitLab
2. A webhook triggers (Commit trigger) the automation processes on Jenkins

3. Jenkins Master selects a Slave for execution
4. Auto-preparing the Scenario, purging old build and clone repository again
5. Build and package based on repository descriptor and upload to k8s for build and test environment
6. The Scenario is prepared to make executions
7. Run tests and get the results
8. A robot create the log report of the tests results and publish them on Jenkins
9. An image is built and published on the Docker Hub repository

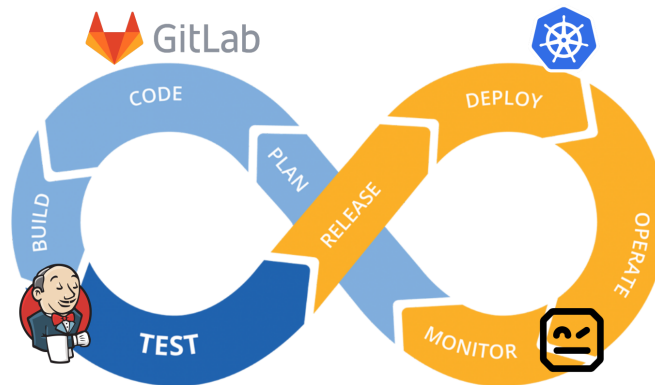






FIGURE 4: CI/CD TOOLS USED IN SCENARIO 1 FLUIDOS

 GitLab	https://about.gitlab.com/	VCS
 Jenkins	https://www.jenkins.io/	CI/CD
 ROBOT FRAMEWORK	https://robotframework.org/	Logs Robotization
 docker hub	https://hub.docker.com/_/registry	Container Registry (Own Registry)


	https://kubernetes.io/	Test Environment
---	---	------------------

TABLE 1 : SCENARIO 1 CORE TOOLS

3.4.2 Scenario 2 (Lightweight Alternative)

In this section, a lightweight alternative is proposed, to save installation, administration and maintenance costs and to have a plug and play scenario, which allows teams to focus their efforts on software development. This scenario has some advantages over the previous one but also some drawbacks such as the lack of privacy or flexibility.

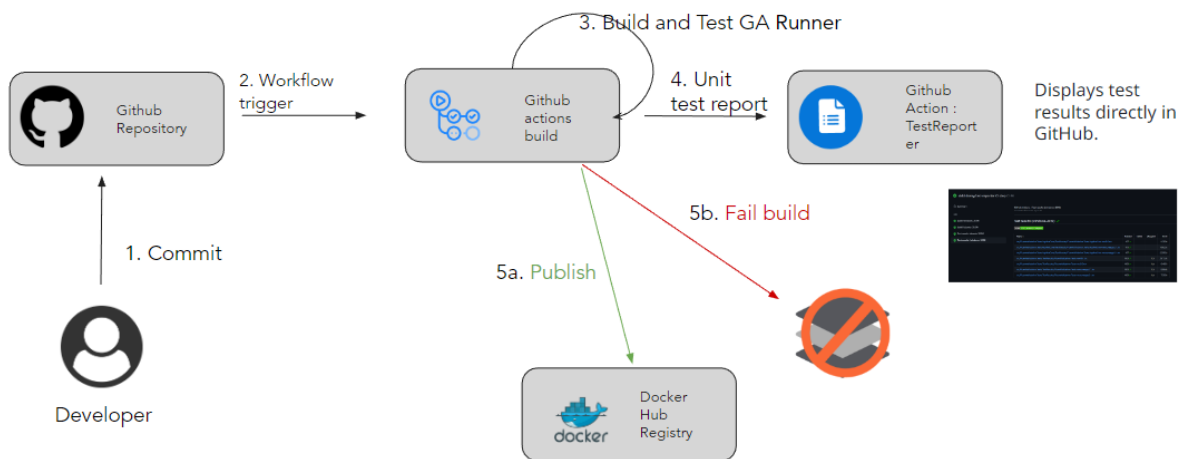


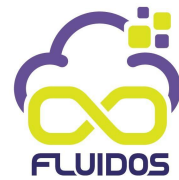
FIGURE 5: SCENARIO 2 WORKFLOW

2. The developer commit code to the project code repository in Github
3. A webhook triggers(Commit trigger)
4. A Build is prepared in a Github Actions Runner
5. Unit testing is displayed directly in Github with GA Test Reporter
6. a) If the build works correctly, it will be published in Docker Hub Registry
b) If the build or testing fail, the artefact is not ready to publish

3.4.2.1 Tools selected

- Version Control System: Github

In this scenario, Github will be used as VCS. The fundamental difference with Gitlab from the previous scenario is that it will not be necessary to do any type of installation and it will not entail a waste of resources because the free functionality with public repositories will be used.



- Continuous Integration: Github Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

GitHub Actions goes beyond just DevOps and lets run workflows when other events happen in a repository.

GitHub provides Linux, Windows, and macOS virtual machines to run workflows, or can be self-hosted runners in a own data centre or cloud infrastructure.

About the pricing, Github has no restrictions for free Organisations that have public repositories and brings 2000 free minutes of build time per month. If these limits are exceeded, the idea is to consider hosting our own runners.



- Logs and Report: Test Reporter (Github Actions)

A Github Action that displays test results from testing frameworks directly in GitHub.

- Parses test results in XML or JSON format and creates nice report as Github Check Run.
- Annotates code where it failed based on message and stack trace captured during test execution.
- Provides final conclusion and counts of passed, failed and skipped tests as output parameters.

- Container Registry: Docker Hub (Public Registry)

The standard registry for Docker and Kubernetes, Highly scalable central space for storing and distributing container images. It provides secure image management and a fast way to pull and push images with the right permissions and without administration overhead or resource costs. The only problem with public registries is that we do not have full control over their actions and that they can get expensive if you need multiple private images.

 GitHub	https://github.com
 GitHub Actions	https://github.com/features/actions




	https://github.com/marketplace/actions/test-reporter
	https://hub.docker.com/_/registry

TABLE 2 : SCENARIO 2 CORE TOOLS

3.4.3 Optional Components

Additional components can be added to complete and improve the Scenarios.

- **SonarQube (code Quality)**

Reducing the risk of software development within a very short amount of time, by constantly scanning code for potential bugs, bad coding practices, code redundancy, etc. is an important component in a CI/CD environment. It detects bugs in the code automatically and alerts developers to fix them before rolling it out for production. Early detection can reduce potential defects in later phases.

SonarQube is an open-source platform offering static analysis, continuous inspection and reviewing of code. It offers support for numerous programming languages as well as integration with Jenkins pipelines. In FLUIDOS it could perform security and quality assurance tests.

- **Keycloak (identity, access management)**

Keycloak is an open-source Identity and Access Management (IAM) tool having a licence with an Apache License 2.0. It streamlines the authentication process for applications and IT services. Keycloak offers a broad set of features, like SSO, authentication and authorization, social login, multifactor authentication, and centralised user management. The purpose of Keycloak is to ensure that the right people in an organisation have appropriate access to resources. With Keycloak, one can secure services with a minimum of time and add authentication to applications.

In FLUIDOS CI/CD Keycloak acts as the SSO point for most of the tools. By using Keycloak there is no need to manage and create user accounts to each tool but centrally controlling and issuing user groups and access details.

- **Nginx (reverse proxy)**

In FLUIDOS, the NGINX could be used as a *reverse proxy* in front of the FLUIDOS CI/CD platform. A *reverse proxy* is a proxy server in a private network, which is located between the firewall and the back-end servers. Its role lies in redirecting and *forwards client requests* (e.g.,



web browser) to the appropriate back-end destination in order to ensure the smooth flow of network traffic between clients and servers (NGINX, 2022a).

- **Portainer (monitoring)**

In FLUIDOS, Portainer could be containerized and deployed on a cloud server and in its Home screen, Portainer can list all the Docker hosts/servers of the environment. Moreover, an overview of the status of each Docker host is displayed briefly on the dashboard. Monitoring and management options are also offered on a per-container basis for each Docker host.

- **Kubernetes (test environments)**

Kubernetes could help control the deployment of the applications that are being developed to build a functional scenario ready for make executions, unit testing, integration testing in which any state of the application can be built.





4 SECURITY PRACTICES FOR THE CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY PLATFORM AND DEVELOPMENT & TESTING ENVIRONMENT

A set of security features has been considered and incorporated in the CI/CD solution to protect both the CI/CD infrastructure and services and the deployed project's artefacts. The following paragraphs describe in detail these provided security assets.

Encryption - Secure communication over Hypertext Transfer Protocol Secure (HTTPS)/Transport Layer Security (TLS)

Access to the offered services is secured with HTTPS to protect the connections of the users to the deployed applications. Specifically, the continuous integration (Jenkins), the management and the artefacts repository (JFrog Container Registry) have been secured with HTTPS, allowing a secure experience from a client's side. Data sent using HTTPS is secured via Transport Layer Security protocol (TLS), which provides three key layers of protection:

- **Encryption** - encrypting the exchanged data to keep it secure from eavesdroppers. That means that while users are using an HTTPS secured service, nobody can "listen" to their conversations, track their activities across multiple pages, or steal their information.
- **Data integrity** - data cannot be modified or corrupted during transfer, intentionally or otherwise, without being detected.
- **Authentication** - proves that users communicate and send data to the intended service. It protects against man-in-the-middle (MitM) attacks and builds user trust.

The OpenSSL library that provides an open-source implementation of the TLS protocol has been used to generate the private keys, certificate signing requests, SSL certificates (self-signed or Certificate Authority (CA)-signed) and for certificate format conversion.

Additionally, the connection to the servers that will be used for the deployment of the project's artefacts, including the testing, staging and production environments, has been also secured with HTTPS. Access to the Docker daemon socket is protected by enabling TLS, allowing Docker to be reachable through the network in a safe manner. From the server side, connections from clients authenticated by a certificate (self-signed or signed by a CA) are allowed to the servers in which the project's artefacts will be deployed. From the client side, clients can only connect to these servers with a certificate that can again be self-signed or signed by a CA.

Hard disk encryption at rest of the CI/CD infrastructure servers

All hard disks mounted to the servers (virtual machines) that are used for the deployment of the FLUIDOS CI/CD solution are encrypted. Disk encryption at rest ensures that files are always stored on disk in an encrypted form. The files only become available to the operating system and applications in readable form while the system is running and unlocked by a





trusted user. An unauthorised person looking at the disk contents directly, will only find garbled random-looking data instead of the actual files, securing the actual data stored in cases that the hard disk or server is lost, stolen, or discarded after its end-of-life.

User authentication for the CI/CD services

The services offered in the FLUIDOS CI/CD solution, such as the continuous integration (Jenkins) and the artefacts repository (Artifactory) are secured using user authentication. The integration of an existing Lightweight Directory Access Protocol (LDAP) server or Active Directory implementation is possible.

Firewall protection of the CI/CD and Development & Testing Environment infrastructure

The security of the infrastructure used by the CI/CD platform and Development & Testing Environment is ensured by proper firewalling, controlling the allowed connections to the servers used for the deployment of services and project's artefacts. Specifically, one of the most popular and flexible open-source Linux firewall software is used for this purpose, iptables. A set of iptables rules has been defined and configured to the different servers to block some of the common network attacks (SYN flood attacks, Smurf attacks, attacks, attacks by malfunctioning ICMP packets and other forms of Denial of Service (DoS attacks)). The default policy is to drop incoming, outgoing or forwarded packets from any source to any destination, unless there is a specific rule set to allow a particular communication.

SSH key-based authentication to the infrastructure

SSH access to administer and manage the CI/CD platform servers has been configured to use only key-based authentication. This is a more secure alternative in comparison to the most commonly used password authentication. Although passwords are sent to the server in a secure manner, they are sometimes not complex or long enough to be resistant to repeated, persistent attackers. Modern processing power combined with automated scripts make brute forcing a password-protected account very possible. Thus, SSH public key authentication in which we generate and store a pair of cryptographic keys and then configure the servers to recognize and accept the generated keys is a more secure approach.





5 CONCLUSIONS

This deliverable has provided technical documentation for all the developments and activities conducted in WP9 FLUIDOS Platform Development Environment Setup. The CI/CD environment set up is presented here, together with the tools chosen to make up the CI/CD stack of FLUIDOS. Moreover, the DevOps methodology and CI/CD practices that will be used during the development and testing phases of the FLUIDOS components are described in detail.

Regarding the Development and Testing environment, a production level CI/CD pipeline could be developed able to integrate distributed deployment and multiple branches of the FLUIDOS components. Moreover, the CI/CD workflow that the project's development team will employ to deploy, test, and integrate software components is introduced.

As each FLUIDOS component has different needs, it falls upon their Product Owners to choose the CI/CD scenario that accomplishes them and the most suitable tools to make it possible.





REFERENCES

- [1] Altameem, E. (2015). Impact of agile methodology on software development. *Computer and Information Science*, 8(2). <https://doi.org/10.5539/cis.v8n2p9>
- [2] Atlassian. (2021, April 6). What is version control: Atlassian Git Tutorial. Atlassian. Retrieved September 15, 2022, from <https://www.atlassian.com/git/tutorials/what-is-version-control>
- [3] Atlassian. (n.d.). What is continuous integration. Atlassian. Retrieved September 8, 2022, from <https://www.atlassian.com/continuous-delivery/continuous-integration>
- [4] GitHub Inc. (n.d.). Where the world builds software. GitHub. Retrieved September 15, 2022, from <https://github.com/>
- [5] Jenkins. (n.d.). Jenkins. Retrieved September 8, 2022, from <https://www.jenkins.io/>
- [6] Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*, Addison-Wesley.
- [7] Let's Encrypt. (n.d.). Let's Encrypt. Retrieved September 20, 2022, from <https://letsencrypt.org/>
- [8] SonarQube. (n.d.). Code quality and code security. SonarQube. Retrieved September 8, 2022, from <https://www.sonarqube.org/>
- [9] Zhao, Y., Serebrenik, A., Zhou, Y., Filkov, V., & Vasilescu, B. (2017). The impact of continuous integration on other software development practices: A large-scale empirical study. 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). <https://doi.org/10.1109/ase.2017.8115619>

