Deliverable D1.5

DevOps methodology and CI/CD strategy for EMERALD-v1

Editor(s):	Gorka Benguria Elguezabal, Iñaki Etxaniz (TECNALIA)
Responsible Partner:	TECNALIA Research and Innovation
Status-Version:	Final - v1.0
Date:	30.04.2024
Type:	R
Distribution level (SEN, PU):	PU

Project Number:	101120688
Project Title:	EMERALD

Title of Deliverable:	D1.5 DevOps methodology and CI/CD strategy for EMERALD-v1
Due Date of Delivery to the EC	30.04.2024

Work package responsible for the Deliverable:	WP1 - Concept and methodology of EMERALD
Editor(s):	Gorka Benguria Elguezabal, Iñaki Etxaniz (TECNALIA)
Contributor(s):	
Reviewer(s):	Franz Deimling (FABA), Cristina Martínez (TECNALIA), Juncal Alonso (TECNALIA)
Approved by:	All Partners
Recommended/mandatory readers:	WP1, WP2, WP3, WP4, WP5

Abstract:	Initial version of the description of the DevOps strategy and setting up of CI/CD environments. Details the DevOps infrastructure and tools to support the continuous integration and deployment phases. Describes the CI/CD strategy for the integration of the EMERALD Framework.
Keyword List:	DevOps, CI/CD, Integration, Container, Environment, Releases
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 4.0 International (**CC BY-SA 4.0 DEED** https://creativecommons.org/licenses/by-sa/4.0/
Disclaimer	Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. The European Union cannot be held responsible for them.

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	15.04.2024	First draft version	TECNALIA
v0.2	17.04.2024	Typos corrected and style polished. Sent for internal QA review	TECNALIA
v0.3	22.04.2024	QA Review	Franz Deimling (FABA)
v0.4	26.04.2024	Addressed all comments received in the Internal QA review	TECNALIA
v1.0	30.04.2024	Submitted to the European Commission	Cristina Martínez (TECNALIA)

Table of contents

List of figures

Terms and abbreviations

CaaS	Certification as a Service
CI	Continuous Integration
CD	Continuous Deployment
CMMI	Capability Maturity Model Integration
CSA or EU CSA	EU Cybersecurity Act
CSP	Cloud Service Provider
DoA	Description of Action
EC	European Commission
GA	Grant Agreement to the project
KPI	Key Performance Indicator
IaC	Infrastructure as Code
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
SW	Software
TRL	Technology Readiness Level

Executive Summary

EMERALD CaaS (Certification-as-a-Service) Framework is built up by several components. These components have different levels of maturity and are developed by different groups. This supposes a challenge when facing the integration of the CaaS Framework as a whole. On the validation side, the project has different pilots, each one with their needs and constrains.

This document describes the DevOps Methodology and CI/CD (Continuous Integration/Continuous Deployment) strategies that are applied for the development of the Certification-as-a-Service (CaaS) Framework to manage the heterogeneity of the components, and to coordinate the different teams in the development and operation sides.

The document audience is the EMERALD participants in charge of the coordination of the development and operation activities. Besides, the document is also focussed to provide information to the rest of partners in the understanding on how the EMERALD CaaS Framework was managed from a DevOps perspective.

The document also includes a pair of annexes that provide extra information about the project, that could help the reader grasp the context of the project. Namely, they include the risks and the milestones defined in the Description of Action (DoA).

This document is the first version of the development and operation coordination approach, that will be applied as a baseline during the first stage of the project. During this first stage, some insights will surely be found. These lessons learned will be described in the next version of the document, and from them some modification of the approach can be derived, that will be documented as well.

1 Introduction

This section introduces the context of the project, the aim and audience of the content and the document structure. This deliverable is the result of task T1.3 *Continuous integration and optimization*, and details the line of action to be followed in the project to achieve a smooth and quick transition among the developers work and the final result: the EMERALD framework deployed in the pilots.

1.1 About this deliverable

From the project mission:

"EMERALD's mission is to provide a user-friendly framework to help stakeholders in the cybersecurity field efficiently manage certifications, enhancing the security and effectiveness of cloud service usage. The proposed EMERALD environment will be the foundation for defining a new service for assisting the certification process that we named Certification-as-a-Service (CaaS)." [1]

To contribute to that mission, this deliverable describes the technical coordination approach that will be followed in the EMERALD project to continuously integrate, update, and validate that framework, here in after referred as **CaaS Framework**. The elements of the approach have been defined considering the EMERALD project context and the planned CaaS Framework, that are detailed in next chapters.

- In this document we describe the two main elements of the technical coordination approach: the DevOps Methodology, and the CI/CD Strategy.

The **DevOps Methodology** in EMERALD will focus on how the development, integration and validation teams will collaborate to build and evolve the CaaS Framework through their planned releases to achieve the evolving requirements considering the EMERALD project resources and constrains.

The **CI/CD Strategy** describes the technical approaches to be applied to continuously integrate and deploy the outcomes from the different development teams to be validated by the involved pipelines. The continuous integration (CI) deals with the integration of the different components that build up the CaaS Framework and the verification of the integrated version so that it is ready for its deployment. The continuous deployment (CD) deals with the deployment of the integration version in the production environment and the required pilot environments for their validation. Within the continuous deployment we will also establish the mechanisms to manage the feedback into new or extended requirements.

This target audience of this document is twofold:

- First of all, the people in charge of DevOps management during the EMERALD project. For them this document should be a guide for managing the integration of the technical outcomes of the development teams, as well as the deployment of the integrated versions for their validation.
- Secondly, this document is also targeted to people that want to understand how the CaaS Framework was developed. For them it will also provide some resources that will be useful to extend or customize the CaaS to their needs.

This document is a first version for the DevOps methodology and CD/CI strategy for EMERALD. Another version will be released in month 18, where the findings on its application will be detailed, as well as any modification applied during that period (M1-M18) or for the upcoming period (M19-M36).

1.2 Document Structure

The document is organized in two main sections:

- DevOps Methodology
- CI/CD Strategy

The DevOps Methodology section, Section 2, explains the foundations for the approach that will be used to coordinate the development and operation in other to continuously integrate, validate and manage feedback to improve the CaaS Framework, considering the needs of the EMERALD pilots and the contributions of the development teams.

The CI/CD Strategy section, Section 3, is split in two sections: the CI Strategy and the CD Strategy. The CI Strategy describes the principles that will guide the integration of the different components of the CaaS Framework prior to its disposal to the pilots. The CD Strategy describes the approach to provide the integrated versions of the CaaS Framework to the pilots for their validation, as well as the mechanisms to collect that validation feedback for the continuous improvement of the upcoming CaaS Framework versions.

Finally, Section 4 presents the main conclusions of the document.

In addition, the document includes two annexes (*APPENDIX A: Project Risks and impact in the DevOps methodology* and *APPENDIX B: Project Milestones from DoA*) that will support the understanding of the document purpose and structure, as well as the technical implementation details of some specific assets of the DevOps infrastructure used.

2 DevOps Methodology

This section describes the process and the lifecycle that will be applied in EMERALD to coordinate the development and operation teams during the project, to help in the achievement of the project objectives.

The section also presents the challenges and risks faced by the EMERALD DevOps Methodology. From these challenges, we will define some top-level goals for the DevOps Methodology to be applied. Then, we present the process to be applied, whose main tasks are specified. Finally, the software lifecycle is presented, that describes how the process is applied over time.

2.1 Context

EMERALD presents some challenges and risk that should be managed during the CaaS Framework development. The Description of Action (DoA) of EMERALD [2] includes some risks (see *APPENDIX A: Project Risks and impact in the DevOps methodology* for more details and an extended list) that are relevant for the definition of the DevOps Methodology:

- Users experience low usability.
- EMERALD components are not able to be fully integrated.
- The implementation does not cover all the use cases.
- Underestimation of effort needed to complete activities.
- Technology changes require significant redesign of the EMERALD architecture.
- A partner fails to meet the obligations and becomes non-performing or even defaulting.

Apart from these risks, enumerated in the DoA, there are other challenges to be considered during the DevOps methodology elaboration:

- We are aiming a TRL7 [3] System/process prototype demonstration in an operational environment (integrated pilot system level).
- We have different components with different requirement sets, different teams, and different agendas.
- We have fixed milestones (*APPENDIX B: Project Milestones from DoA*) at project level that should be achieved.
- The CaaS Framework will be deployed as a Service. That implies to integrate and test all the components in a production grade service environment.
- Some pilots, due to internal policies, may require deploying the framework for validation internally.

2.2 Goals

Based on the risk and challenges established, the DevOps Methodology we are aiming for should have the following characteristics.

- **Release based**: We need a release-based methodology because the DoA has stated an iterative approach for the CaaS Framework, where at least three releases are going to be provided. Therefore, a minimum of three versions are expected, with some intermediate versions that can be motivated by other milestones of the project.
- **Manage the feedback**: The project aims a TRL7 outcome. That implies that at the end of the project, apart for being finished, the system must be validated in real life environments. The project has several validation cycles planned, and the methodology should keep track of the issues raised during these activities to make sure they are managed.

- **Manage the aimed component set**. To keep track of the integration of all the components, it is necessary to have a clear idea of which the components of the EMERALD project are, which are integrated, and which are not.
- **Keep requirement traceability**. We will use the requirements as the basis for the validation of the different components of the CaaS Framework, as well as the framework as a whole. Therefore, there should be a traceability of the DevOps activities with the requirements set. The requirements will be managed in Gitlab and reported in D1.3 "EMERALD solution architecture v1" to be submitted in M12.
- **Manage the environments**. The EMERALD project will be required to manage several environments during the DevOps activities. The project envisions at least two environments: integration and production, but additional ones can be managed on demand. Integration environment is focussed on providing debugging support for the developers and verification means for the project. Production environment is focussed in providing a validation platform for the project. Additional ones can be created at any point by the pilots themselves or by the DevOps team, based on specific needs, and lasting for a variable timeframe.
- **Integrate as soon as possible**. The update requests from the diverse development teams should be promoted into the integration environment as soon as possible. This will help to mitigate some of the risks identified, such as the effort underestimation and the partner withdrawal.

2.3 Processes

To select the set of processes to use to support the coordination between the development and pilots, we have a wide range of standards and other kinds of references to choose from. From the standards side, there is no normative DevOps standard that can be used as a basis. The most approximate elements that can be found are standards for software development and for service operation maturity evaluation, such as:

- CMMI v1.3 [4] where we can find 22 process areas.
- ISO 15504 [5] where we can find 48 processes.
- ITIL [6] where we have 34 management practices.

From these standard references, interesting process categories can be extracted, such as architecture, monitoring, release, issue, validation, deployment, infrastructure, integration, requirements, training, management, improvement, configuration …

Focussing on the DevOps process – as stated above – there is no standard DevOps process as such, nor even a sound "de facto" standard. A review of the literature shows that, even if there are some propositions about a DevOps process model in [7], [8], there is no sound common reference [9], [10], [11], [12].

For the definition of the processes to be executed in the EMERALD DevOps Methodology, we will take as a starting point the DevOps cycle that is used in many publications when describing DevOps [13], [14], [15], [16], [17]. Figure 1 shows the basic structure of the DevOps cycle, shared along those publications. This cycle is also present throughout grey literature [18].

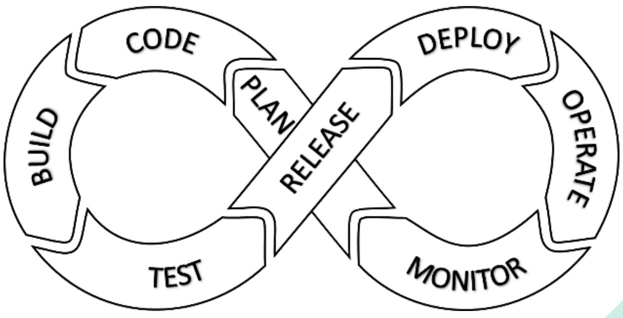


Figure 1. DevOps Cycle

We will take these processes as a basis for the DevOps Methodology, adapting them to the specificities of the EMERALD project. For each process, we provide below a brief description and the expected inputs and outputs.

Plan: During the planning process we will manage to collect the integration and deployment needs of the project. Those needs will be prioritised and planned as issues in the CaaS Framework repository in GitLab.

- Inputs:
    - Requirements.
    - Feedback from pilots' validation.
    - EMERALD Architecture.
    - External service requirements.
- Outputs:
    - Issues linked to the requirements.

Code: During the coding process, the integration of the different components will be encoded with a container orchestration code (aka choreography). In addition, code for the environment and side services can be developed when needed.

- Inputs:
    - Components from the different development teams in a package repository (implemented in Artifact).
- Outputs:
    - Choreography code and configuration for the integration environment.
    - Infrastructure as code (IaC) for the environments.
    - IaC for testing the environment.
    - Code for the services.

Build: The building process will be automated. It will take the choreography and will deploy the CaaS Framework in the integration environment.

- Inputs:
    - Packages from the components.
    - Choreography code and configuration for the integration environment.
- Outputs:
    - Integration environment.
    - CaaS Framework in the integration environment.

Test: The test process will focus on the integration testing. It will cover the development of the necessary tests as well as the monitoring mechanism to verify the behaviour of the framework in the long term.

- Inputs:
    - CaaS Framework in the integration environment.
- Outputs:
    - Integration tests.
    - Integration test results.
    - Integration monitoring procedures.
    - Integration monitoring results.

Release: During the planned release milestones, as well as when other releases require it, we will create a tag in the CaaS Framework repository.

- Inputs:
    - Release request.
- Outputs:
    - Integration tests.
    - Integration test results.
    - Monitoring procedures.
    - Monitoring results.

Deploy: The deploy process will be automated. It will take the choreography and will deploy the tagged CaaS Framework in the production environment, and in any other required pilot environment.

- Inputs:
    - Packages from the components.
    - Choreography code, configuration for the production and pilot environments.
- Outputs:
    - Production environment and pilot environments, if required.
    - Tagged CaaS Framework in the production and pilot environments, if required.

Operate: The validation process will be carried out by separate teams in WP5. Therefore, this process will be focussed on the validation management.

- Inputs:
    - Tagged CaaS Framework in the production and pilot environments, if required.
- Outputs:
    - Feedback from pilots' validation.

Monitor: The monitoring of the integration environment will be adapted in this stage to cover the production and pilots' environments.

- Inputs:
    - Tagged CaaS Framework in the production and pilot environments, if required.
- Outputs:
    - Monitoring procedures.
    - Monitoring results.

2.4 Lifecycle

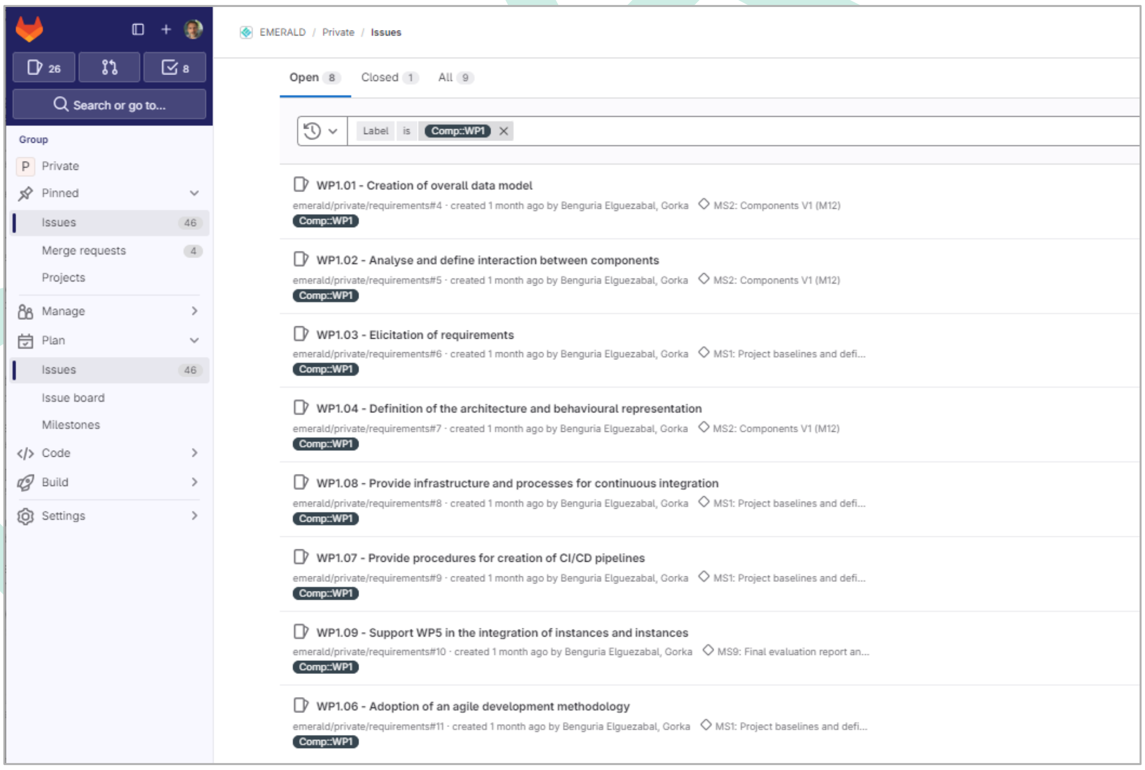
There are several applicable lifecycles in software development [19], [20], but considering the following characteristics of the EMERALD project:

- There are different components with different requirement sets, different teams, and different agendas.
- There are fixed milestones (see *APPENDIX B: Project Milestones from DoA*) at project level that should be achieved.

We decided to apply an iterative process as stated in the DevOps lifecycle (see Figure 1). The DevOps iterations will be continuous, whenever the DevOps team receives integration and release requests from the development teams. Besides those requests, the DevOps team will perform iterations every week, focussing on the environment's setup, integration tests, monitoring mechanisms, and monitoring results.

We will use two mechanisms to document the tasks to be performed (e.g., implement monitoring service in k8sv) in the context of the DevOps activities: issues, and merge requests.

- **Issues**[1] will be used as the primary mechanism for documenting tasks that involve some effort on the part of the DevOps team. Every task will be documented in an issue inside the affected repos under the DevOps group in the project Gitlab repository (see Figure 2).
- **Merge requests**[2] will be used mainly to allow development teams to upgrade the components versions in the CaaS framework choreograph, and other minor changes that do not require significant DevOps team interaction (see Figure 3).



WP1.01 - Creation of overall data model
emerald/private/requirements#4 · created 1 month ago by Benguria Elguezabal, Gorka MS2: Components V1 (M12)
Comp::WP1

WP1.02 - Analyse and define interaction between components
emerald/private/requirements#5 · created 1 month ago by Benguria Elguezabal, Gorka MS2: Components V1 (M12)
Comp::WP1

WP1.03 - Elicitation of requirements
emerald/private/requirements#6 · created 1 month ago by Benguria Elguezabal, Gorka MS1: Project baselines and defi...
Comp::WP1

WP1.04 - Definition of the architecture and behavioural representation
emerald/private/requirements#7 · created 1 month ago by Benguria Elguezabal, Gorka MS2: Components V1 (M12)
Comp::WP1

WP1.08 - Provide infrastructure and processes for continuous integration
emerald/private/requirements#8 · created 1 month ago by Benguria Elguezabal, Gorka MS1: Project baselines and defi...
Comp::WP1

WP1.07 - Provide procedures for creation of CI/CD pipelines
emerald/private/requirements#9 · created 1 month ago by Benguria Elguezabal, Gorka MS1: Project baselines and defi...
Comp::WP1

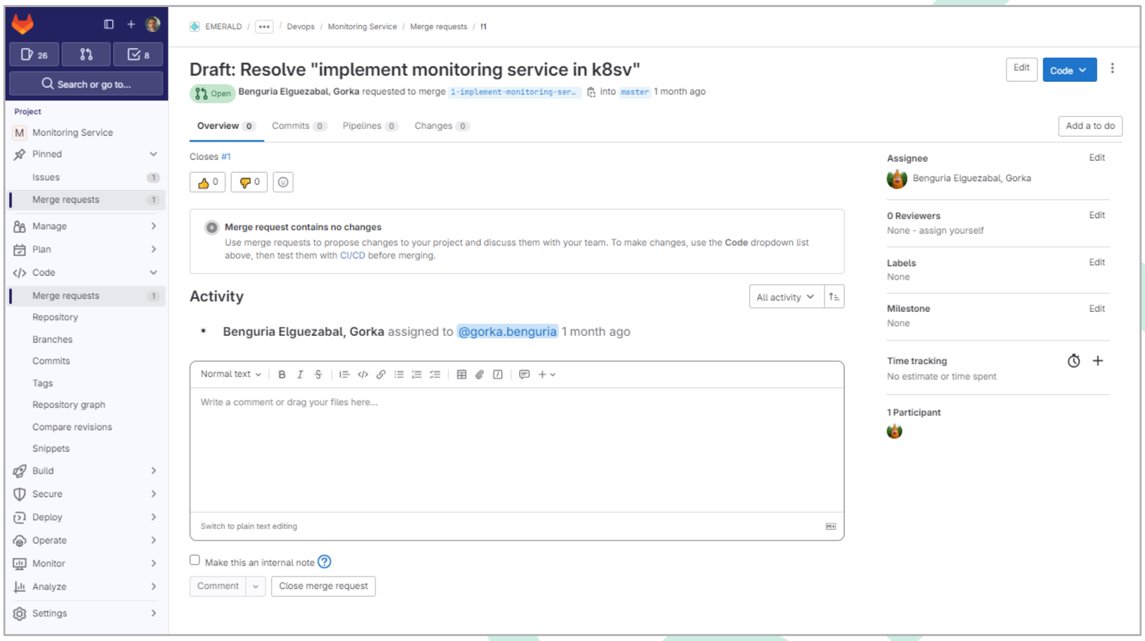
WP1.09 - Support WP5 in the integration of instances and instances
emerald/private/requirements#10 · created 1 month ago by Benguria Elguezabal, Gorka MS9: Final evaluation report an...
Comp::WP1

WP1.06 - Adoption of an agile development methodology
emerald/private/requirements#11 · created 1 month ago by Benguria Elguezabal, Gorka MS1: Project baselines and defi...
Comp::WP1

Figure 2. List of Issues related to Concept & methodology in EMERALD

[1] https://docs.gitlab.com/ee/user/project/issues/
[2] https://docs.gitlab.com/ee/user/project/merge_requests/

In both cases, the task has to be related with some requirement(s), to keep track of their implementation in the DevOps activities. In the case of the issues, they will be linked with the requirements using the "linked requirements" mechanism provided by Gitlab. In the case of the merge requests, the merge request will be linked to an offline activity in order to minimize the barriers to verify changes in the integration or production environments, on behalf of the development teams.



Draft: Resolve "implement monitoring service in k8sv"

Open Benguria Elguezabal, Gorka requested to merge `1-implement-monitoring-ser...` into `master` 1 month ago

Closes #1

Merge request contains no changes
Use merge requests to propose changes to your project and discuss them with your team. To make changes, use the **Code** dropdown list above, then test them with CI/CD before merging.

Activity

- **Benguria Elguezabal, Gorka** assigned to @gorka.benguria 1 month ago

Write a comment or drag your files here...

Switch to plain text editing

☐ Make this an internal note

Assignee Edit
Benguria Elguezabal, Gorka

0 Reviewers Edit
None - assign yourself

Labels Edit
None

Milestone Edit
None

Time tracking
No estimate or time spent

1 Participant

Figure 3. A Merge Request related to Concept & methodology in EMERALD

Tasks will be managed in an agile approach, with some flexibility due to the need to quickly update components at the request of the development teams. Regular tasks will enter the DevOps lifecycle as requests in the task backlog (i.e., as issues in the DevOps repository).

On every iteration we will perform an internal review to:

- Review the status of completion of the tasks worked out during the last iteration.
- Choose the new set of tasks to be carried out during the next iteration.

The status of the DevOps tasks will be visible at the DevOps group level in GitLab.

The execution of each task may involve one or more planned processes, with the exception of the monitoring process that will be automated and continuous. These processes are: Code, Build, Test, Release, Deploy, and Operate. These processes are not expected to be carried out in every task. Depending on the nature of the task, it may involve all or some of them, and the effort in each task will vary.

3 CI/CD Strategy

This section will provide details on the strategies and technical approach outlined in this first stage of the EMERALD project. We will focus specifically on the integration and deployment activities.

3.1 CI Strategy

For the integration of the outcomes of the development teams in the CaaS Framework, we will apply the following technological approaches:

- Components packaged as containers.
- Environment defined with Infrastructure as code (IaC).
- Progressive verification.
- Integration automation.

3.1.1 Container-based

Container technology has proved to be a very good approach to aggregate components from different teams. Besides, if used appropriately, it also provides de facto scalability and resilience when we use container orchestration technologies, such as Kubernetes [21] or Docker Swarm [22]. In addition, the usage of the container technology promotes decoupling of the architecture which provides some benefits over monolithic architectures [23], [24].

EMERALD will prioritize container images as the default packaging technology for its components. In case some components cannot be deployed as containers, IaC and service approaches will be prioritized as backup strategies.

As a container technology, we will use the Docker ecosystem to build and share images. For image building we will support both *Docker* and *Docker Compose*. A component of the EMERALD architecture may include one or more *dockerfiles* [25] that will be used to build the images that will be used to deploy the CaaS Framework. In some cases, the building process may require some orchestration. In those cases, *Docker Compose* or *Custom Scripting* can be used as well.

To support both scenarios as part of the DevOps activities, we will provide resources and support for the automation of building such images. We will use different technologies including:

- **Docker shared runners** that support *Docker in Docker* (dind) [26] technology
- **Kubernetes group runners**
- **Docker machine shared runners** that support dind technology

As an IaC technology, OpenTofu [27] and Ansible [28] will be favoured. Both technologies are open-source and facilitate knowledge sharing and latter distribution of the resources.

Finally, in case there are some components that cannot be physically deployed in production and must be consumed as a service, the OpenAPI Specification will be promoted [29].

Regarding the strategy with respect to the packaging on behalf of the DevOps team, it is planned to work as follows:

- Provide example packaging approaches starting from a *Docker* or a *Docker Compose* specification. The examples will be specific to Gitlab framework used in the EMERALD project. The examples will include:
  - dgitlab-ci.yml for *Docker* and gitlab-ci.yml for *Docker Compose*.
  - Example of a dockerfile.

  - Example of a docker compose file.
  - Readme file with indications on how to integrate the component.
- These examples will include guides on how to integrate it on the components that are stored in the Gitlab framework used in the EMERALD project.
- If necessary, WP1 will provide support in the integration of the gitlab-ci, and in the development of *Docker* and *Docker compose*. This will be managed through a merge request from the interested party, that will be documented in an issue related to a project requirement.

3.1.2 Environments with IaC

The environments that support the CaaS Framework integration and validation will be developed following an IaC approach with state of the practice open-source tools.

The integration environment has been developed following this approach. For that, a project has been created in the private GitLab of EMERALD (/devops/opentofu-k8sv). This project creates a four-node Kubernetes cluster over vSphere platform. The project includes instructions to replicate the deployment on another vSphere platform if necessary. In addition, it uses a reusable set of Ansible playbooks to configure EMERALD Kubernetes that have been applied in this case, but could be applied on other nodes with little or no customization.

The IaC is configurable through the modification of two templated yaml files:

- Base OpenTofu hosts: (/blob/master/base_opentofu_hosts.yaml.erb), this controls the initial creation of the virtual machines and their configuration by Ansible.
- Ansible host: (/blob/master/Ansible_hosts.yaml.erb), this configures the machines using some Ansible-playbooks.

The same approach will be followed for the instantiation of other environment and resources, so that we can replicate them in case we need to do so in latter stages of the project.

In the same way that any other activity in the DevOps team, the IaC development will be documented in an issue related to an EMERALD project requirement and implemented through a merge request.

3.1.3 Progressive Verification

The verification of the added and updated components of the CaaS Framework is an important aspect to ensure the secure evolution of the platform during the project. The verification will be covered by a set of integration tests that will be automated.

The verification activities will have two main focusses:

- Establish the means to check the health of the components.
- Define more complex integration tests based on developer requests and validation feedback.

In the first stage of the project, as components are added, procedures should be implemented to check the health of the components. These mechanisms will be used during the integration tests performed after the update of each component, as well as during continuous monitoring.

As the project advances, and based on requests from the developers and feedback from the pilots, additional integration tests will be added.

The strategy in EMERALD will be to document verification-focussed activities as issues linked to requirements. Implementing means to check the health of a component may require implementing parts in the choreography and parts in the component itself. More complex integration may require implementing specific components to generate the activity required to verify such complex integration scenarios.

3.1.4 Automation

DevOps activities will focus on the automation of all the activities related to the evaluation of components as they are updated by the developers. The main focuses of automation will be:

- **Update the integration platform** as the developers update the components.
- **Run the integration test** as the platform is updated.
- **Update the monitoring mechanism** to measure the health of the CaaS Framework in the long term.

The strategy will be to use *GitLab Agent* for Kubernetes, implemented in the devops repository (devops/gitlab-agent-k8sv). It monitors the CaaS Framework repository, and every change detected there will be translated into the aimed environments. It will allow us to deploy new component versions directly, without integration testing. This is done to speed up the feedback to the development team. Integration tests will be started at a later stage, using a Gitlab runner.

The monitoring mechanism will be updated following the same approach as the CaaS Framework. That is, we will use the Gitlab Agent for Kubernetes for this purpose as well.

3.2 CD Strategy

For the deployment of the CaaS Framework to be evaluated by the project and the pilots, the following technological approaches will be applied:

- Releases
- Documentation
- Environment defined with IaC
- Deployment automation

3.2.1 Releases

The project will follow a versioning system with three mayor releases:

- v1.0.0 - First release of the EMERALD components in month 18
- v2.0.0 - Second release of EMERALD components in month 30
- v3.0.0 - Final release of EMERALD integrated audit suite in month 34

Additional releases are expected between those mayor releases, as the project advances and the CaaS Framework is validated. Versions v0.x.x will be created during the first iterations of the project before the month 12. Versions v3.x.x are also expected based on the feedback of the last validation activities.

The strategy for the releases will start with a petition from an EMERALD partner. Basically, the release consists of moving the content in the integration branch to the production branch. To perform this process an issue will be created describing the request and the purpose. From that issue a new draft merge request will be created over the production branch. That will create a new working branch, where we will move the integration version that we want to move into production. The integration version to be deployed should have been successfully verified with the integration tests, otherwise we will notify the risk before proceeding. After that, we change remove the draft status on the merge request, that will enable the merge action over the production branch. We perform the Merging that will update the production version. Finally, we communicate the change to the EMERALD project.

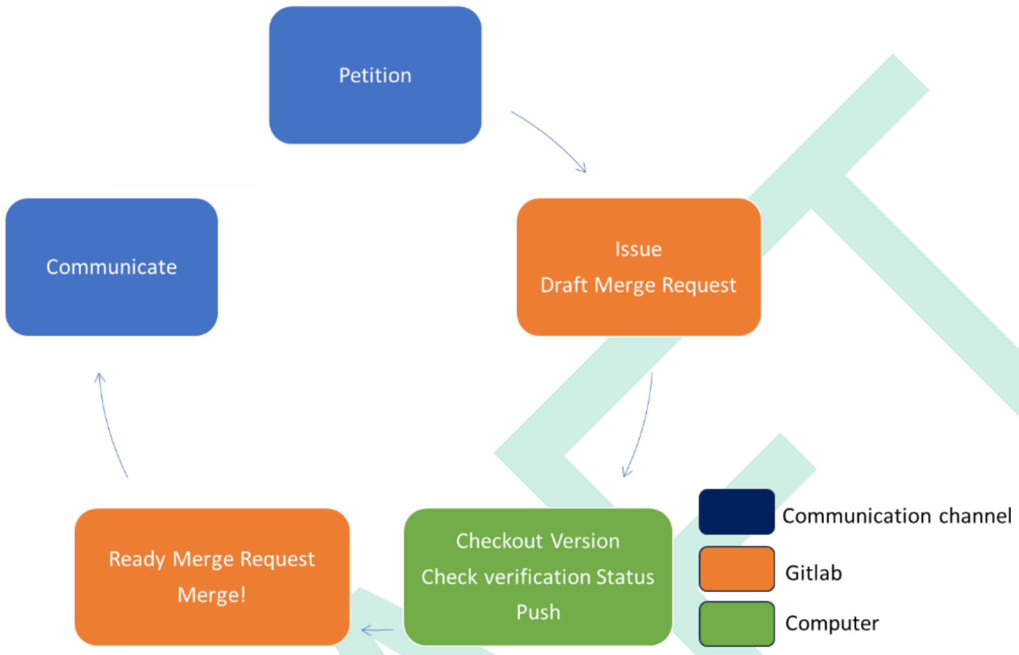


Figure 4. A Merge Request mechanism to produce a new release in EMERALD

3.2.2 Documentation

The documentation regarding the CaaS Framework will be generated as part of the deployment to production. The focus of the documentation will be in the deployment and configuration of the pilots.

The strategy for the documentation will be related with the changes in the IaC related with the environment and the CaaS Framework itself.

3.2.3 Environments with IaC

The production environment will be deployed using IaC, with the same approach used for the integration environment. Besides, the DevOps team will support the generation of additional environments, if needed. For example, environments on pilot's premises due to privacy or legal restrictions.

The strategy for the production environment IaC will be similar to the strategy with the integration IaC, i.e., as any other activity in the DevOps team, it will be documented in an issue related to an EMERALD project requirement, and implemented through a merge request.

3.2.4 Automation

The DevOps strategy will work towards the automation of all activities related to the creation of releases and their deployment in the production environment. The main focuses of automation will be:

- **Deploying specific releases** to specific environments.
- **Updating the monitoring mechanism** to measure the long-term health of the CaaS Framework.

The automation strategy during deployment will be similar to that of the integration. We will use *GitLab Agent for Kubernetes* to translate new releases on the production environment. For the pilot environments, we will leave it up to the pilot owners to decide the procedure for updating their respective environments.

4 Conclusions

This document presents the DevOps methodology and the CI/CD strategy to be applied to in the EMERALD project.

The methodology used customizes the commonly used DevOps lifecycle to the characteristics and constrains of the EMERALD project. This lifecycle consists of the following steps: Plan, Code, Build, Test, Release, Deploy, Operate, and Monitor. The goals of the defined methodology are: to be release-based; manage feedback; manage components; keep traceability; manage the environments; and integrate as soon as possible. In this line, we have presented the main customised processes, as well as a tailored approach to iterate, so that we prioritise speed of integration over other elements.

In the strategy part we have described the technical approaches that we will implement to support the EMERALD project needs. In this sense, we will leverage some technologies and state of the practice DevOps resources, such as:

- Configuration management with IaC
- Gitlab features with respect to:
    - Issues
    - Git workflows (branches and merge requests)
    - Automation with GitLab ci
    - Documentation
- Releases with containers
- Container orchestration technologies

This document is the first version of the DevOps approach for EMERALD, where we have established the baselines of the DevOps work. In a year's time, we will provide an updated version with an in-depth description of the lessons learnt, possible modifications to the process and details of the IaC developed.

5 References

[1] EMERALD Consortium, “Home page,” [Online]. Available: https://www.emerald-he.eu/. [Accessed April 2014].

[2] EMERALD Consortium, “EMERALD - Annex 1 - Description of Action - GA 101120688,” 2022.

[3] ISO, “ISO 16290:2013, Space systems — Definition of the Technology Readiness Levels (TRLs) and their criteria of assessment,” 2013.

[4] CMMI Dev, “CMMI for Development, Version 1.3,” Software Engineering Institute (SEI), Ed., 2010.

[5] International Organisation for Standardization (ISO/IEC), “ISO/IEC 15504-1:2004 Information Technology – Process Assessment – Part 1: Concepts and Vocabulary,” 2004.

[6] AXELOS, “ITIL Foundation,” Stationery Office Books, Norwich, England, 2019.

[7] J. A. V. M. K. Jayakody and a. W. M. J. I. Wijayanayake, “Process Improvement Framework for DevOps Adoption in Software Development,” in *2023 International Research Conference on Smart Computing and Systems Engineering (SCSE), IEEE, Jun. 2023. doi: 10.1109/scse59836.2023.10214992*, 2023.

[8] I. Bucena and M. Kirikova, “Simplifying the DevOps Adoption Process,” in *BIR Workshops, pp. 1–15*, 2017.

[9] R. d. Feijter, “Towards the adoption of DevOps in software product organizations: A Maturity Model Approach,” Master’s Thesis, 2017.

[10] S. Badshah, A. A. Khan and B. Khan, “Towards Process Improvement in DevOps: A Systematic Literature Review’,” in *Proceedings of the Evaluation and Assessment in Software Engineering, EASE ’20. ACM, Apr. 2020. doi: 10.1145/3383219.3383280*.

[11] R. Amaro, R. Pereira and M. M. da Silva, “Capabilities and Practices in DevOps: A Multivocal Literature Review,” in *IEEE Trans. Softw. Eng., vol. 49, no. 2, pp. 883–901, Feb. 2023, doi: 10.1109/tse.2022.3166626*.

[12] M. Gasparaitė and S. Ragaišis, “Comparison of devops maturity models,” in *IVUS 2019. Proceedings of the International Conference on Information Technologies Kaunas, Lithuania, April 25, 2019, CEUR-WS. org, 2019, pp. 65–69*.

[13] R. T. Yarlagadda, “DevOps and its practices,” in *Int. J. Creat. Res. Thoughts IJCRT ISSN, pp. 2320–2882*, 2021.

[14] A. Colantoni, L. Berardinelli and M. Wimmer, “DevopsML: Towards modeling devops processes and platforms,” in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, pp. 1–1*, 2020.

[15] R. Amaro, R. Pereira and M. M. da Silva, “DevOps Metrics and KPIs: A Multivocal Literature Review,” in *ACM Comput. Surv., Mar. 2024, doi: 10.1145/3652508*.

[16] A. V. Jha et al., “From theory to practice: Understanding DevOps culture and mindset,” in *Cogent Eng., vol. 10, no. 1, p. 2251758*, 2023.

[17] H. R. Kadaskar, “Unleashing the Power of Devops in Software Development,” in *Int. J. Sci. Res. Mod. Sci. Technol., vol. 3, no. 3, pp. 01–07*, 2024.

[18] Wikipedia, “DevOps toolchain,” 20 Jan 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=DevOps_toolchain&oldid=1197449470. [Accessed Apr 2024].

[19] A. M. Davis, E. H. Bersoff and a. E. R. Comer, “A strategy for comparing alternative software development life cycle models,” in *IEEE Trans. Softw. Eng., vol. 14, no. 10, pp. 1453–1461, doi: 10.1109/32.6190*, 1988.

[20] A. Mishra and D. Dubey, “A comparative study of different software development life cycle models in different scenarios,” in *Int. J. Adv. Res. Comput. Sci. Manag. Stud., vol. 1, no. 5*, 2013.

[21] “Production-Grade Container Orchestration,” [Online]. Available: https://kubernetes.io/. [Accessed April 2024].

[22] “Swarm mode overview, Docker Documentation,” [Online]. Available: https://docs.docker.com/engine/swarm/. [Accessed April 2024].

[23] M. Kalske and others, “Transforming monolithic architecture towards microservice architecture,” Univ. Hels., 2017.

[24] J.-P. Gouigoux and D. Tamzalit, “From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture,” in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, France: IEEE, Apr. 2*.

[25] “Dockerfile reference | Docker Docs,” [Online]. Available: https://docs.docker.com/reference/dockerfile/. [Accessed April 2024].

[26] “Use Docker to build Docker images | GitLab,” [Online]. Available: https://docs.gitlab.com/ee/ci/docker/using_docker_build.html. [Accessed April 2024].

[27] “OpenTofu,” [Online]. Available: https://opentofu.org/. [Accessed April 2024].

[28] “Homepage | Ansible Collaborative,” [Online]. Available: https://www.Ansible.com/. [Accessed April 2024].

[29] “OpenAPI Specification - Version 3.1.0 | Swagger,” [Online]. Available: https://swagger.io/specification/. [Accessed April 2024].

[30] EMERALD Consortium, “D7.1 Project Manual and Quality Plan,” 2024.

APPENDIX A: Project Risks and impact in the DevOps methodology

This section makes an analysis of some of the risks defined in the project – initially from Description of Action [2] and then extended in D7.1 [30]] - and the impact that the DevOps Methodology can have to mitigate them. It is important to note that these can evolve as part of the EMERALD *Task 7.2 Quality Assurance & Risk Management.*

Risk n.	Description	Proposed Mitigation Measures
1	EUCS is not ready until 2026.	Not relevant.
2	Incompatibility between OSCAL and EMERALD (data import/export, modelling of security schemes).	Not relevant.
3	Users experience low usability.	WP4 will work in the UI/UX concept. The methodology contains activities to manage feedback from pilots.
4	EMERALD components are not able to be fully integrated.	The methodology should control which components are integrated and which components are not integrated.
5	Data set not sufficient for reaching TRL7 on the evidence collector components.	Not relevant. It should be controlled by the validation activities as part of other work packages.
6	The implementation does not cover all the use cases.	The methodology deployment activities should keep track of the use cases involved.
7	Underestimation of effort needed to complete activities.	The methodological approach promotes short cycles that will help to identify those situations faster. Besides, short cycles will focus on having running versions, and clearer view of what is missing. .
8	Technology changes require significant redesign of the EMERALD architecture.	Integration tests will help to verify the redesigned elements. This will speed up the verification of refactored components as they are changed to the new architecture.
9	A partner fails to meet the obligations and becomes non-performing or even defaulting.	The methodological approach should promote multiple versions to have partial versions (instead of no versions) in that case.
10	Partner heterogeneity: The different organizational and national cultures cause collaboration problems or conflicts in the project consortium	The methodology has been defined at the beginning of the project, and the clarity of the process paves the way for an easier collaboration.
13	Project execution risks: a) key milestones are delayed b) critical deliverables are delayed	The DevOps methodology, bringing together the work of developers, integrators and final users, helps to mitigate the possible delays.
14	Project key technologies, development risks: a) Key technologies or components are not available at the expected time	The DevOps methodology can be easily adapted to cover other development languages or technologies. The DevOps methodology automates the integration of the source code, and thus

	b) development takes longer than expected c) wrong technology base is selected d) lacking consensus on the technological approach between scientific partners	can speed up the deployment of delayed releases to make them available for the Use Cases.
25	Use case implementation is poor	The DevOps methodology will produce three releases, as defined in the project plan, and the successive feedbacks can help making a better final implementation. Also, the IaC approach of the DevOps methodology is a mitigation measure.

APPENDIX B: Project Milestones from DoA

This section includes the project milestone taken from the Description of Action (DoA) of EMERALD [2].

#	Title	Validation	Month
1	Project baselines and definition. Pilot set-up. Certification Graph Schema created.	• Project manual, public website. Defined the dissemination, communication and networking strategy. • Market analysis for EMERALD developed. • Data modelling for EMERALD components and initial design and requirements of the components. • CD/CI methodology for EMERALD defined. • Pilots' definition and evaluation strategy set up.	9
2	First release of the EMERALD components.	• EMERALD overall design specification and architecture • Initial prototypes of the main components of EMERALD.	12
3	First release of EMERALD integrated audit suite. First version of the EMERALD business models and plans, communication and dissemination report.	• Initial prototype of the EMERALD integrated solution with the functionalities implemented at M12. • First versions of the EMERALD business models dissemination and communication reports. • Second version of EMERALD CD/CI methodology.	18
4	First implementation and evaluation of the first release of the EMERALD solution in the pilots.	• First implementation of the first release of the EMERALD tools in the use cases	20
5	Second release of EMERALD components.	• Second version of the EMERALD architecture. • Second releases of the main components of EMERALD.	24
6	Second release of EMERALD integrated audit suite.	• Second prototype of the EMERALD integrated solution with the functionalities implemented at M24.	30
7	Second implementation and evaluation of the second release of the EMERALD solution in the pilots.	• Second implementation of the second release of the EMERALD tools in the use cases.	32
8	Final release of EMERALD integrated audit suite.	• Final prototype of the EMERALD integrated solution with feedback from the second evaluation of the pilots.	34
9	Final evaluation report and impact analysis. Final version of the EMERALD business models and plans,	• Final evaluation and impact analysis from the pilots. • Final versions of the EMERALD business models dissemination and communication reports.	36

	communication and dissemination report.		