EO4EU
Earth Observation Data & Service Accessibility

D4.7 - Release of the integrated framework [Dev & Production level] (a)

Work Package	WP4, EO4EU Data Marketplace Ecosystem
Lead Author (Org)	ENG
Contributing Author(s) (Org)	NKUA, EBOS, NVCR, CINECA, ECMWF
Due Date	30.11.2023
Date	16.11.2023
Version	V1.0

Dissemination Level

X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

Disclaimer

Funded by the European Union

Versioning and contribution history

Version	Date	Author	Notes
0.1	10.10.2023	Piero Scrima (ENG) - Giovanni Barone (ENG)	TOC and v0.1
0.2	24.10.2023	Piero Scrima	CI/CD Process v0.2
0.3	01.11.2023	Giovanni Barone - Lakis Christodoulou (EBOS)	GUI Description v0.3
0.4	06.11.2023	Charalampos Andreou (NKUA)	Integration and Validation Tests chapter
0.5	10.11.2023	Dimitris Tsakalidis (NVCR) - George Domalis (NVCR)	Refinement of Knowledge Graph description
0.6	13.11.2023	Beatrice Chiavarini (CINECA)	Added XR/VR chapter
1.0	14.11.2023	Piero Scrima – Giovanni Barone	Content updates
1.1	15.11.2023	Tolga Kaprol (ECMWF) - Claudio Pisa (ECMWF)	Comments and reviews
1.2	16.11.2023	Piero Scrima – Giovanni Barone	Comments and review addressed
1.3	27.11.2023	Salvatore Marchese (IES)	Internal review
1.4	27.11.2023	Costas Rizogiannis (KEMEA)	Internal review
1.5	28.11.2023	Piero Scrima (ENG), Giovanni Barone (ENG)	Final Version

Terminology

Terminology/Acronym	Description
KG	Knowledge Graph
CI	Continuous integration
CD	Continuous deployment
CFS	Customer Facing Services
ML	Machine Learning
SSO	Single-Sign On
GUI	Graphical User Interface
WFE	Workflow Editor
XR	Extended Reality

Terminology/Acronym	Description
API	Application programming Interface
DSL	Domain Specific Language
EO	Earth Observation

Table of Contents

List of Figures

Executive Summary

The document "D4.7 Release of the Integrated Framework" gives a targeted and focused overview of the functionality of the initial version of the EO4EU platform. It also includes the methodology and the tools used for the collaborative development of the project. This document also gives an overall picture of high-level integration and validation testing methodologies. The document serves as quick reference and guide to the platform's capabilities, offering insights into development tools, continuous integration/deployment (CI/CD) practices, user interface design, and workflow management.

1 Introduction

This report documents the first release of the EO4EU integrated framework. The objectives of this document are two: firstly, to facilitate an in-depth understanding and utilization of the newly developed framework and the internal systems of the project, and secondly, to provide a detailed account of the processes and methodologies that have culminated in this release.
The document is a reference for navigating the platform, exploring the code developed and the processes produced so far, while furthermore it gives insight on the testing and validation activities that led to the release of the current system.

In chapter 2 a generic introduction of code repositories and CI/CD methodology is provided.

In chapter 3 the main tools used in the development and collaboration process are presented and described. Finally, the work produced for the EO4EU project through these tools is shown.

In chapter 4, the testing procedure of high-level components and validation methods that are following deployment in test, staging and production environments will be described.

In chapter 5, an end-user perspective of the results ensuring that users can easily navigate and leverage the tools and functionalities offered by the platform is provided.

2 Code Repository and CI/CD Process

In the current landscape of software development, efficient and streamlined integration processes and the related integration tools are essential components of any successful project. Centralized and distributed source control, artifacts repositories, Continuous Integration (CI) and Continuous Deployment (CD) practices are fundamental in software integration, when dealing with highly scalable modular architectures (Figure 1). Another important development methodology is the so-called DevOps, a combination of development (Dev) and operations (Ops), it emphasizes the importance of communication, collaboration and integration between software developers and IT operations. DevOps approach aims to shorten the systems development lifecycle, ensuring high software quality and aligning with CI and CD practices. Finally, lately with the spread of Machine Learning, another development methodology linked to Machine Learning Operations (MLOps) algorithms has emerged. MLOps is a set of practices that aims to deploy and maintain machine learning models in production, reliably and efficiently. In these scenarios, it is required to quickly move from code to running components, from development to deployment. These practices and tools have emerged as indispensable pillars in an era defined by agility, collaboration, and relentless innovation.
For these reasons, EO4EU employs the use of these methodologies for its development, also using software that supports their use such as GitLab as the main code repository, GitLab CI/CD for the automated build and deployment pipeline, and Docker registry as the artifact repository seamlessly integrated into the GitLab Repository.

This document presents (chapter 3) the integration and collaboration tools used within the scope of EO4EU, providing an overview of their function and showing the specific instance of EO4EU. Moreover, it describes the current project, available in the GitLab repository, and the general process adopted in CI/CD pipeline, showing an example taken from a project which leverages a CI/CD script. The integration tools and CI/CD practices described here are crucial in ensuring that the EO4EU user interface, detailed in Chapter 5, is always up-to-date and functioning optimally. Furthermore, they lay the foundation for the comprehensive integration and validation tests covered in Chapter 4.

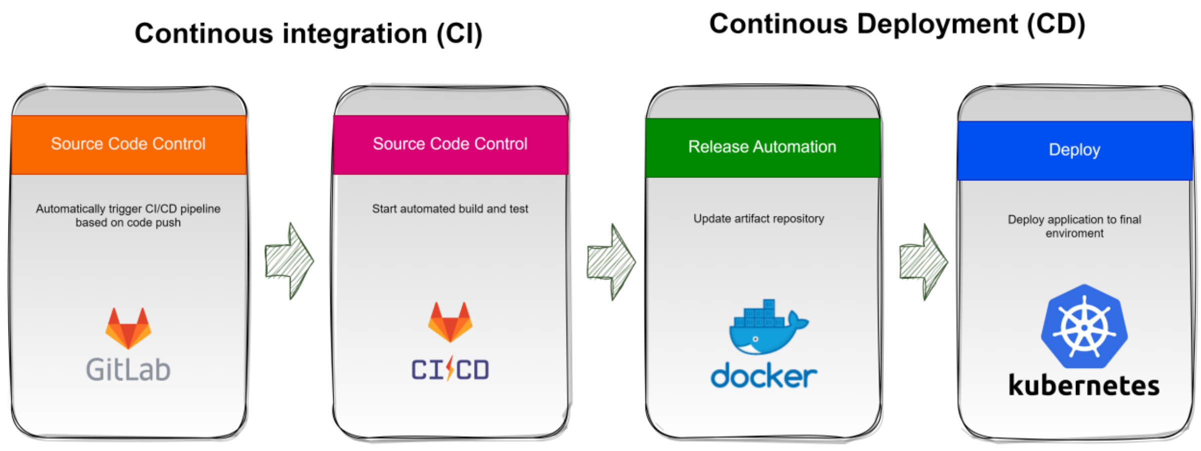

Continous integration (CI)
Continous Deployment (CD)
Source Code Control
Automatically trigger CI/CD pipeline based on code push
GitLab
Source Code Control
Start automated build and test
CI/CD
Release Automation
Update artifact repository
docker
Deploy
Deploy application to final enviroment
kubernetes

Figure 1. CI/CD schema

3 Continuous Integration (CI), Delivery of Development Operations (DevOps), and Machine Learning Operations (MLOps)

This chapter presents some of the tools used in EO4EU software development, such as GitLab. Moreover, it describes how these tools were used in this first part of the project. MLOps will be better integrated in the next phase of the project and therefore its implementation will be addressed in the next version of this document.

3.1 Continuous Integration (CI)

The EO4EU project relies on the Continuous Integration, which is a software development practice that involves regularly merging code changes from multiple contributors into a shared repository. The primary goal is to detect integration issues early and ensure that the software remains in a functional state throughout its development. CI includes automating the build, test, and deployment processes to maintain code quality and streamline collaboration.
The most important continuous integration aspects are:

- Automated Testing: Code changes are subjected to automated tests, including unit tests, integration tests, and regression tests.
- Frequent Integration: Developers integrate their code frequently, often multiple times a day, to catch integration issues early.
- Shared Repository: All developers work on a common codebase, allowing for easy collaboration and conflict resolution.
- Automated Build and Deployment: The CI system automates the process of building the application and deploying it to a testing environment.

Delivery of Development Operations
Furthermore, the EO4EU project follows a DevOps which consists of a set of practices that aim to unify software development (Dev) and IT operations (Ops) by emphasizing collaboration, automation, and communication between teams. It seeks to streamline the software delivery pipeline and improve the efficiency and reliability of the deployment process.

Most important DevOps aspects are:

- Collaboration: DevOps encourages close collaboration between development, operations, and other stakeholders throughout the software lifecycle.
- Automation: Automation tools are used to automate deployment, provisioning, monitoring, and other repetitive tasks.
- Continuous Delivery: Building on CI, DevOps promotes continuous delivery by automating the deployment process, allowing for frequent and reliable software releases.
- Feedback Loops: Continuous monitoring and feedback mechanisms are established to identify and address issues promptly.

3.2 Machine Learning Operations

In addition, MLOps is an extension of DevOps principles to machine learning workflows like in the proposed WFE and ML models deployment. It focuses on efficiently deploying, monitoring, and maintaining machine learning models in production environments.

Most important MLOps aspects are:

- Version Control for Models: Like code, machine learning models and their associated artifacts (datasets, pre-processing scripts) are versioned.
- Automated Model Deployment: Automation is applied to deploy machine learning models into production environments.
- Model Monitoring: Continuous monitoring of model performance and drift is crucial to ensure ongoing accuracy and reliability.
- Feedback Loop: Data collected from production is used to improve and retrain models over time.
- Reproducibility: MLOps emphasizes the reproducibility of model training, ensuring that models can be rebuilt and retrained as needed.

Continuous Integration, DevOps, and MLOps play crucial roles in the design, development, deployment, and integration of various software components within the EO4EU platform system while their use imply different benefits, such as:

Continuous Integration: Ensures that individual components are integrated and tested frequently, identifying and resolving integration issues early.

1. DevOps: Facilitates seamless collaboration between development, operations, and other teams, automating deployment and monitoring processes.
2. MLOps: Allows for the efficient deployment and management of AI/ML models, ensuring their reliability and performance in production.

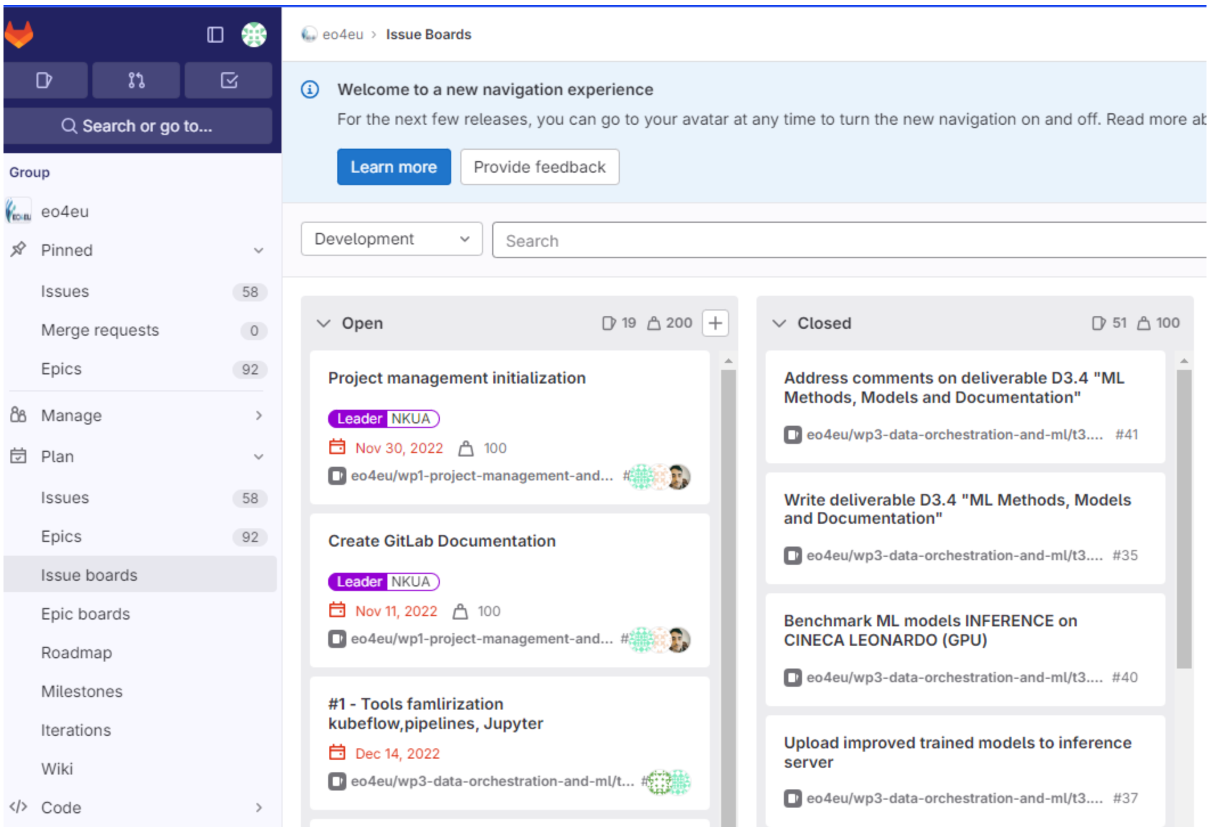
In the next subsection, tools and practices applied in the current development of the EO4EU platform are described.

3.3 Gitlab repository

GitLab, launched as an open-source platform in 2011, has rapidly become a staple in the realm of distributed version control, largely due to its comprehensive suite of features that cater to collaborative software development. Its robustness lies in its ability to handle source code management while facilitating collaborative workflows among geographically dispersed development teams, such as those found in the EO4EU project.
Gitlab not only offers Git repository management but seamlessly integrates various tools and services, improving the development flow and software lifecycle. This includes integration with project management tools, continuous integration systems and third-party services, which helps streamline development processes within EO4EU.

Gitlab provides an issue tracking system that allows developers to create, assign, comment and track project issues. It is useful for bug management, feature requests, and collaboration between team members.
In EO4EU, Gitlab has been adopted also to support task management and planning, precisely because it allows to integrate project management activities with development activities and to keep track of them, as can be seen in Figure 2.
Issue tracking allows the creation of activity issues and the assignment of activities to specific users. The activities can also be labelled with comments, and it is possible to follow their status. The GitLab instance for tracking activities is reachable via the address:

https://gitpcomp.di.uoa.gr/


eo4eu > Issue Boards

Welcome to a new navigation experience

For the next few releases, you can go to your avatar at any time to turn the new navigation on and off. Read more ab

Learn more | Provide feedback

Development | Search

Group: eo4eu — Pinned: Issues 58, Merge requests 0, Epics 92; Manage; Plan: Issues 58, Epics 92, Issue boards, Epic boards, Roadmap, Milestones, Iterations, Wiki; Code

Open — 19, 200

- Project management initialization — Leader NKUA — Nov 30, 2022 — 100 — eo4eu/wp1-project-management-and...
- Create GitLab Documentation — Leader NKUA — Nov 11, 2022 — 100 — eo4eu/wp1-project-management-and...
- #1 - Tools famlirization kubeflow,pipelines, Jupyter — Dec 14, 2022 — eo4eu/wp3-data-orchestration-and-ml/t...

Closed — 51, 100

- Address comments on deliverable D3.4 "ML Methods, Models and Documentation" — eo4eu/wp3-data-orchestration-and-ml/t3.... #41
- Write deliverable D3.4 "ML Methods, Models and Documentation" — eo4eu/wp3-data-orchestration-and-ml/t3.... #35
- Benchmark ML models INFERENCE on CINECA LEONARDO (GPU) — eo4eu/wp3-data-orchestration-and-ml/t3.... #40
- Upload improved trained models to inference server — eo4eu/wp3-data-orchestration-and-ml/t3.... #37

Figure 2. EO4EU issue board

One of the most important features of Gitlab, essential for the EO4EU project, is the possibility of using a fully integrated CI/CD platform, which will be described later in this document.

3.4 EO4EU Gitlab projects

During the initial activities of EO4EU project, a Gitlab environment has been set which centralizes the storage of the code developed during the project. Having the appropriate credentials issued by the EO4EU system administrators, it is possible to access the repository via the following link:

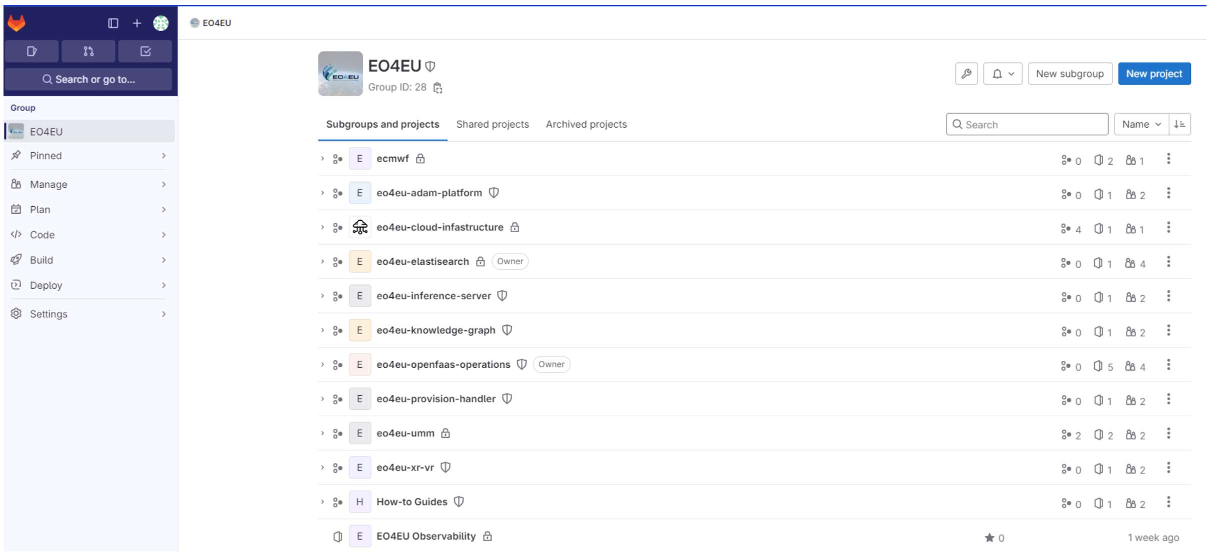
https://git.apps.eo4eu.eu/

The portal is organized by dividing the projects in groups, which for this project is just EO4EU, and nested subgroups. In the first level of subgroups, it is possible to distinguish the following subgroups:

- EO4EU
  - EO4EU cloud infrastructure
  - How To Guides
  - EO4EU umm
  - Custom Facing Services
  - EO4EU Elasticsearch
  - EO4EU Observability
  - EO4EU-XR-VR
  - EO4EU-openfass-operations

- EO4EU-adam-platform
- EO4EU-knowledge-graph
- EO4EU-inference-server

In Figure 3, the list of subgroups can be seen in the GitLab EO4EU website.

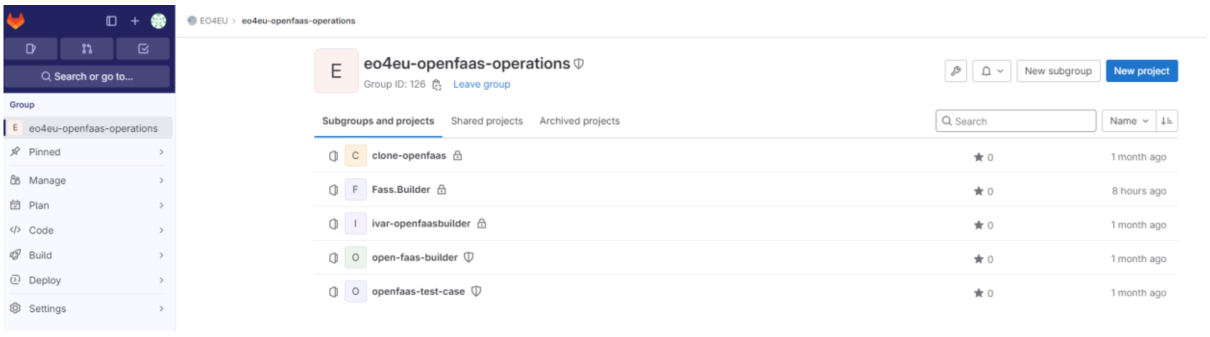


EO4EU
Group ID: 28

E	ecmwf	0	2	1
E	eo4eu-adam-platform	0	1	2
	eo4eu-cloud-infastructure	4	1	1
E	eo4eu-elastisearch Owner	0	1	4
E	eo4eu-inference-server	0	1	2
E	eo4eu-knowledge-graph	0	1	2
E	eo4eu-openfaas-operations Owner	0	5	4
E	eo4eu-provision-handler	0	1	2
E	eo4eu-umm	2	2	2
E	eo4eu-xr-vr	0	1	2
H	How-to Guides	0	1	2
E	EO4EU Observability	★ 0		1 week ago

Figure 3. EO4EU GitLab groups

A subgroup can have in turn other subgroups as shown in Figure 4 for the eo4eu-openfaas-operations group.

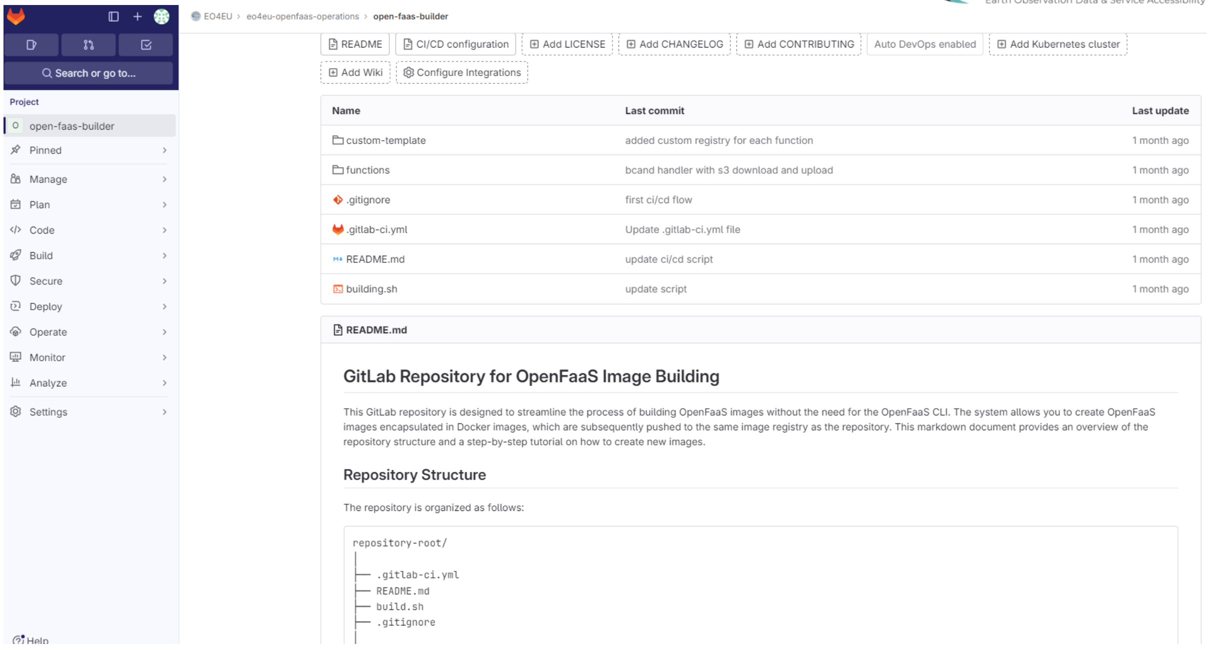


eo4eu-openfaas-operations
Group ID: 126

C	clone-openfaas	★ 0	1 month ago
F	Fass.Builder	★ 0	8 hours ago
I	ivar-openfaasbuilder	★ 0	1 month ago
O	open-faas-builder	★ 0	1 month ago
O	openfaas-test-case	★ 0	1 month ago

Figure 4. EO4EU Gitlab openfaas subgroups

Each of these subgroups contains one or more projects which are described on the first page of the repository as can be seen in Figure 5.



Name	Last commit	Last update
custom-template	added custom registry for each function	1 month ago
functions	bcand handler with s3 download and upload	1 month ago
.gitignore	first ci/cd flow	1 month ago
.gitlab-ci.yml	Update .gitlab-ci.yml file	1 month ago
README.md	update ci/cd script	1 month ago
building.sh	update script	1 month ago

README.md

GitLab Repository for OpenFaaS Image Building

This GitLab repository is designed to streamline the process of building OpenFaaS images without the need for the OpenFaaS CLI. The system allows you to create OpenFaaS images encapsulated in Docker images, which are subsequently pushed to the same image registry as the repository. This markdown document provides an overview of the repository structure and a step-by-step tutorial on how to create new images.

Repository Structure

The repository is organized as follows:

```
repository-root/
│
├── .gitlab-ci.yml
├── README.md
├── build.sh
├── .gitignore
│
```

Figure 5. Gitlab project readme

3.5 GitLab CI/CD

GitLab stands out for its integrated CI/CD solution, an integral part of GitLab's suite of web application capabilities. EO4EU leverages this robust platform to improve project integration and optimize workflow efficiency. In this section, we will look at how GitLab CI/CD works. The following section will be dedicated to the specific projects hosted within the EO4EU GitLab repository.

GitLab's CI/CD architecture is based on the interaction of several fundamental elements: runners, jobs, and pipelines, which together coordinate the automation of the software life cycle, from development to deployment. Each project on GitLab can have a custom CI/CD configuration, expressed via a file called '.gitlab-ci.yml'. This file contains directives that orchestrate the operations of the CI/CD pipeline, including tasks ranging from compiling code, to running tests, to deploying the application.

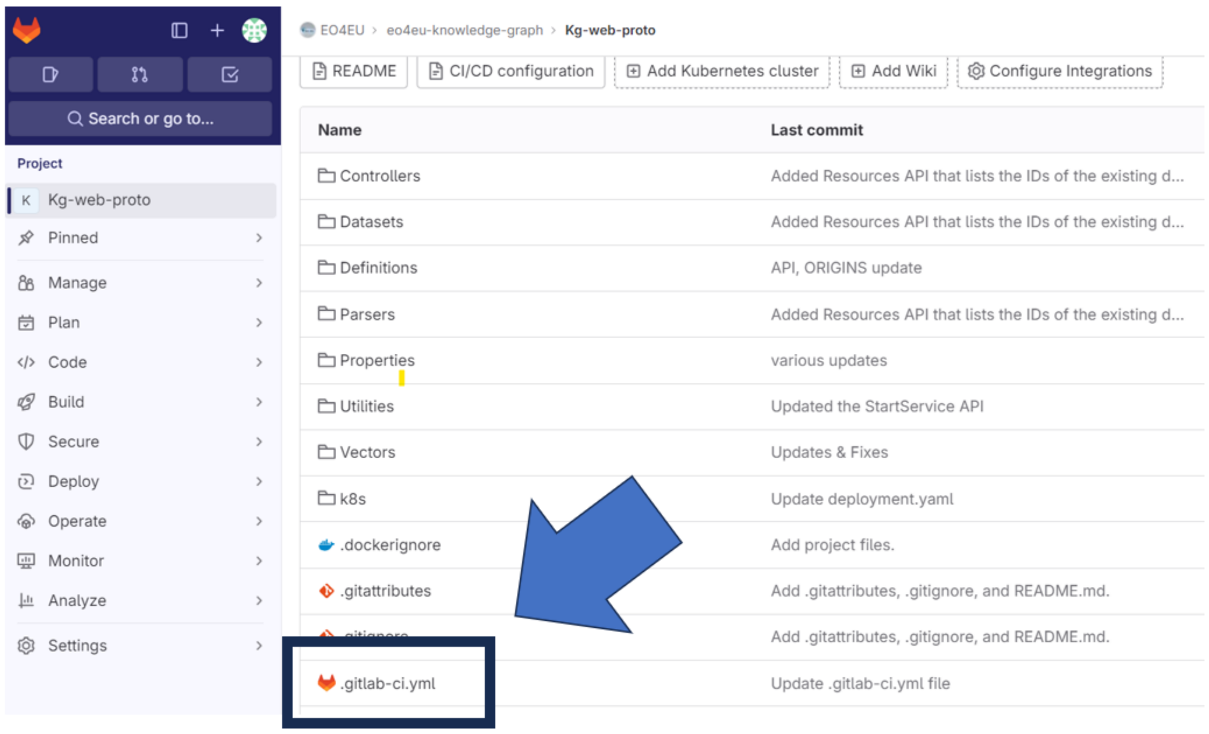
Inside the '.gitlab-ci.yml' file, jobs are organized into phases, forming a structured sequence that determines their execution order. This structured approach ensures that each job only begins upon the successful completion of its predecessor, thus maintaining the integrity of the construction process. User can also declare environment variables within this file to designate specific conditions under which jobs will run.

The activation of a CI/CD pipeline occurs in response to any code push or merge request, thanks to the configurations present in the '.gitlab-ci.yml' file. Jobs within the pipeline run in isolated environments known as Runners-dedicated agents responsible for running the jobs. GitLab features two types of runners: shared runners and specific runners that users can configure on their own infrastructure. This capability gives users the versatility to tailor the execution environment to the precise needs of their projects.

Additionally, GitLab CI/CD provides reports detailing the results of jobs and pipelines. These reports are critical for tracking pipeline performance and quickly identifying any issues, ensuring continuous integration and deployment processes are as efficient and error-free as possible.

3.6 EO4EU CI/CD pipeline

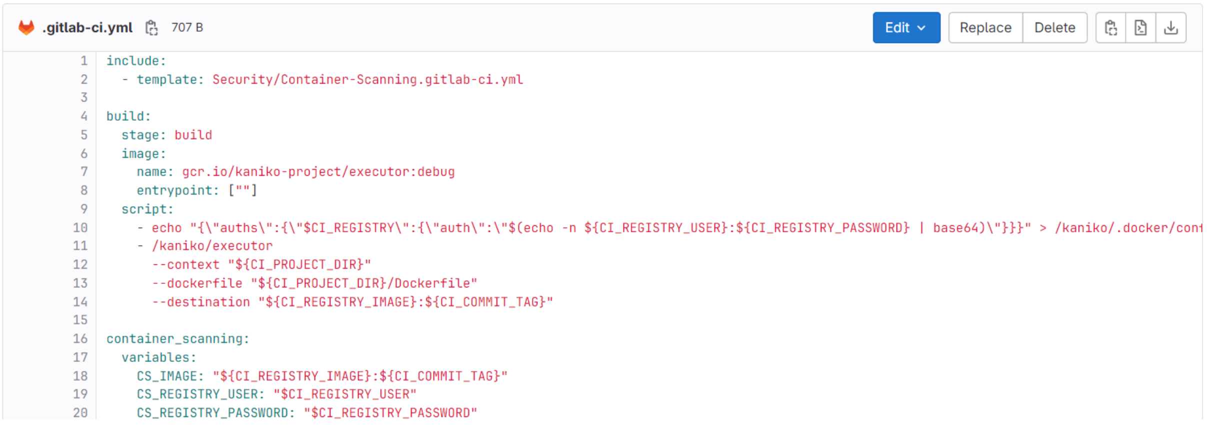
As mentioned above, EO4EU bases its integration strategy on CI/CD mechanisms. To implement the CI/CD pipelines, the tool described in the previous chapter was applied. In this section we will see some concrete examples of how CI/CD is used specifically in EO4EU Gitlab projects.
The GitLab CI/CD pipeline, detailed in this section, not only facilitates rapid development and deployment but also enables automated testing protocols. These protocols are essential for the validation processes described in Chapter 4, ensuring that every release meets our stringent quality standards.
As described previously, the starting point of a CI/CD process is the pipeline descriptor. In GitLab CI/CD this descriptor is specified in a YAML file called .gitlab-ci.yml, as shown in Figure 6, where a screenshot of the list of files belonging to one of the EO4EU Gitlab projects is presented.



Name	Last commit
Controllers	Added Resources API that lists the IDs of the existing d...
Datasets	Added Resources API that lists the IDs of the existing d...
Definitions	API, ORIGINS update
Parsers	Added Resources API that lists the IDs of the existing d...
Properties	various updates
Utilities	Updated the StartService API
Vectors	Updates & Fixes
k8s	Update deployment.yaml
.dockerignore	Add project files.
.gitattributes	Add .gitattributes, .gitignore, and README.md.
.gitignore	Add .gitattributes, .gitignore, and README.md.
.gitlab-ci.yml	Update .gitlab-ci.yml file

Figure 6. .gitlab-ci.yml file

Different Gitlab CI/CD pipelines are employed by EO4EU, however the vast majority have the main objective of creating docker images. In subsequent developments the created Docker images will be deployed in the execution environment automatically. In Figure 7 one of the simplest CI/CD scripts which has been implemented is shown.



.gitlab-ci.yml 707 B

Edit Replace Delete

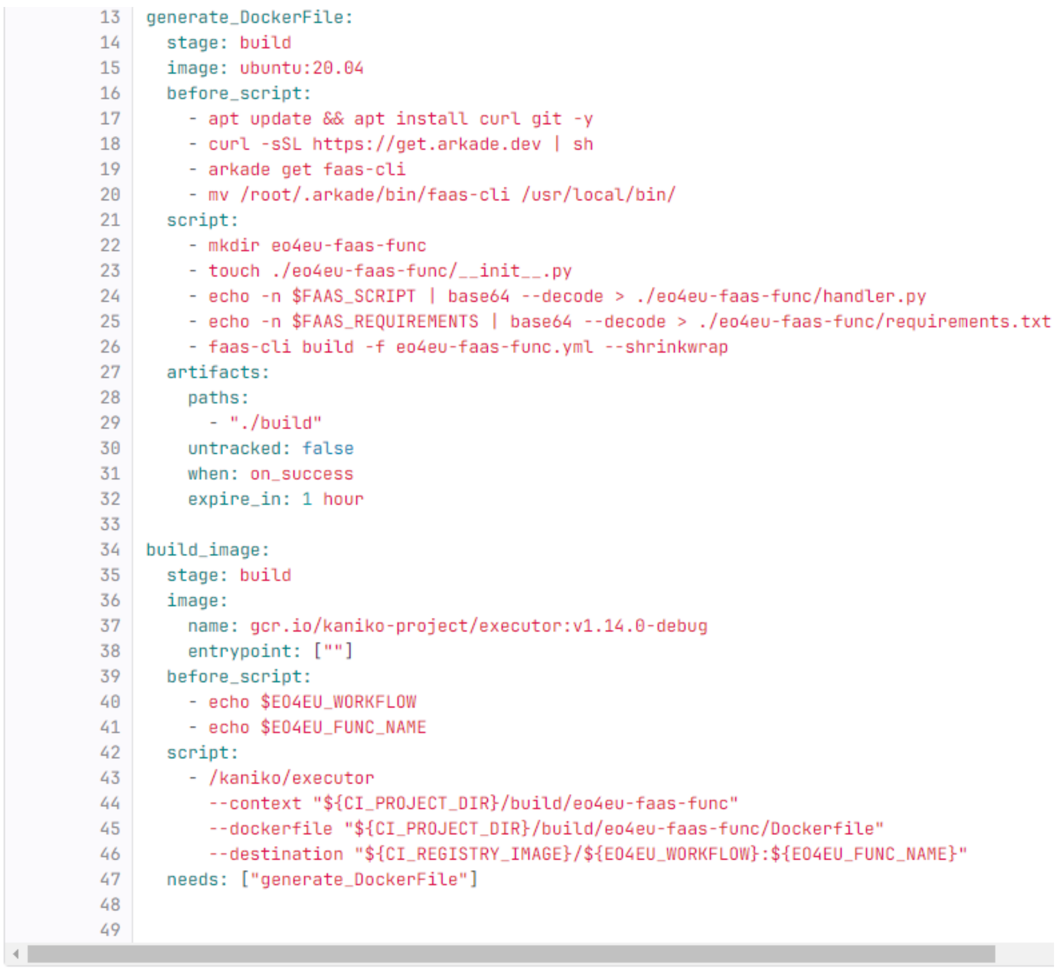
```yaml
include:
  - template: Security/Container-Scanning.gitlab-ci.yml

build:
  stage: build
  image:
    name: gcr.io/kaniko-project/executor:debug
    entrypoint: [""]
  script:
    - echo "{\"auths\":{\"$CI_REGISTRY\":{\"auth\":\"$(echo -n ${CI_REGISTRY_USER}:${CI_REGISTRY_PASSWORD} | base64)\"}}}" > /kaniko/.docker/conf
    - /kaniko/executor
      --context "${CI_PROJECT_DIR}"
      --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
      --destination "${CI_REGISTRY_IMAGE}:${CI_COMMIT_TAG}"

container_scanning:
  variables:
    CS_IMAGE: "${CI_REGISTRY_IMAGE}:${CI_COMMIT_TAG}"
    CS_REGISTRY_USER: "$CI_REGISTRY_USER"
    CS_REGISTRY_PASSWORD: "$CI_REGISTRY_PASSWORD"
```

Figure 7. A simple CI/CD script

The script uses the pipeline to create a Docker image. To build the Docker image in a Kubernetes environment, where GitLab itself is hosted, Kaniko is required. Kaniko is a tool to build container images from a Dockerfile, inside a Kubernetes cluster. Unlike traditional Docker builds, Kaniko does not depend on a Docker daemon, but executes the build in a container environment itself. This approach is particularly useful in GitLab Runners, it avoids the need for privileged containers and mitigates security risks associated with granting Docker daemon access.
Scripts have also been developed for more complex pipelines, as in the case of Openfaas, as shown in Figure 8.



```yaml
generate_DockerFile:
  stage: build
  image: ubuntu:20.04
  before_script:
    - apt update && apt install curl git -y
    - curl -sSL https://get.arkade.dev | sh
    - arkade get faas-cli
    - mv /root/.arkade/bin/faas-cli /usr/local/bin/
  script:
    - mkdir eo4eu-faas-func
    - touch ./eo4eu-faas-func/__init__.py
    - echo -n $FAAS_SCRIPT | base64 --decode > ./eo4eu-faas-func/handler.py
    - echo -n $FAAS_REQUIREMENTS | base64 --decode > ./eo4eu-faas-func/requirements.txt
    - faas-cli build -f eo4eu-faas-func.yml --shrinkwrap
  artifacts:
    paths:
      - "./build"
    untracked: false
    when: on_success
    expire_in: 1 hour

build_image:
  stage: build
  image:
    name: gcr.io/kaniko-project/executor:v1.14.0-debug
    entrypoint: [""]
  before_script:
    - echo $EO4EU_WORKFLOW
    - echo $EO4EU_FUNC_NAME
  script:
    - /kaniko/executor
      --context "${CI_PROJECT_DIR}/build/eo4eu-faas-func"
      --dockerfile "${CI_PROJECT_DIR}/build/eo4eu-faas-func/Dockerfile"
      --destination "${CI_REGISTRY_IMAGE}/${EO4EU_WORKFLOW}:${EO4EU_FUNC_NAME}"
  needs: ["generate_DockerFile"]


```

Figure 8. Openfaas .gitlab-ci.yml

In this case, the pipeline creates the Docker image, but before this step it configures the build environment and executes a tool, faas-cli, to create an instance of function as a service. It is possible to follow the different steps of the pipeline, together with the history, navigating the Gitlab web UI through the sections available from the side menu, as seen in Figure 9.

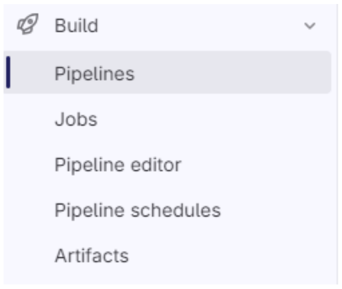
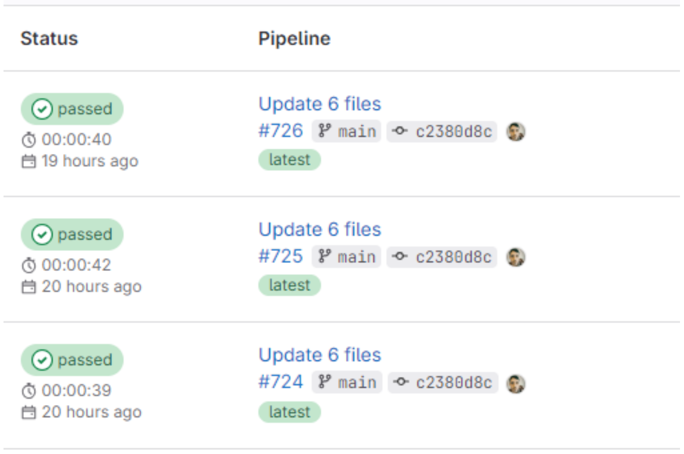


Figure 9. Pipeline menu

The pipeline section lists the pipeline executions with their status, as shown in Figure 10.



Status	Pipeline
passed 00:00:40 19 hours ago	Update 6 files #726 main c2380d8c latest
passed 00:00:42 20 hours ago	Update 6 files #725 main c2380d8c latest
passed 00:00:39 20 hours ago	Update 6 files #724 main c2380d8c latest

Figure 10. Pipeline history

As seen in this section, various pipelines have been designed in EO4EU, which allow the creation of images to be deployed on a Kubernetes environment. The next section describes how the produced images are stored.

3.7 GitFlow

GitFlow is a workflow model designed to improve the management of software projects that use the Git version control system. Created by Vincent Driessen in 2010, GitFlow provides a robust and organized structure that facilitates collaboration between developers and the effective management of different phases of software development.

At the heart of GitFlow there are two main branches: the master branch and the develop branch. The master branch contains the code that is currently in production, while develop branch serves as the basis for the current development of the project. This separation helps to maintain a clear distinction between stable code and code in development.

To manage new features, bug fixes, and release preparations, GitFlow uses a set of supporting branches. The developer that starts working on a new feature has to create a feature branch from the develop branch. This practice isolates new feature development, allowing developers to work on multiple features in parallel without interfering with each other.

Once a feature is complete and tested, the feature branch is merged back into the develop branch. This process ensures that all new features are integrated and tested together before a release preparation.

When the team is ready to release a new version of the software, a release branch is created from develop. This branch is used for final checks and small bug fixes, ensuring that the code is ready to be launched into production. After these phases are completed, the release branch is merged into both master and develop, and the code in master is tagged with a version number, indicating a new official release.

In situations where critical bugs are discovered in production code, GitFlow provides an effective approach to address them via hotfix branches. These branches are created directly from the master branch and are intended to provide quick and straightforward fixes. After the bug is fixed, the hotfix branch is merged into both master and develop, ensuring that the bug is fixed in both the production release and current development.

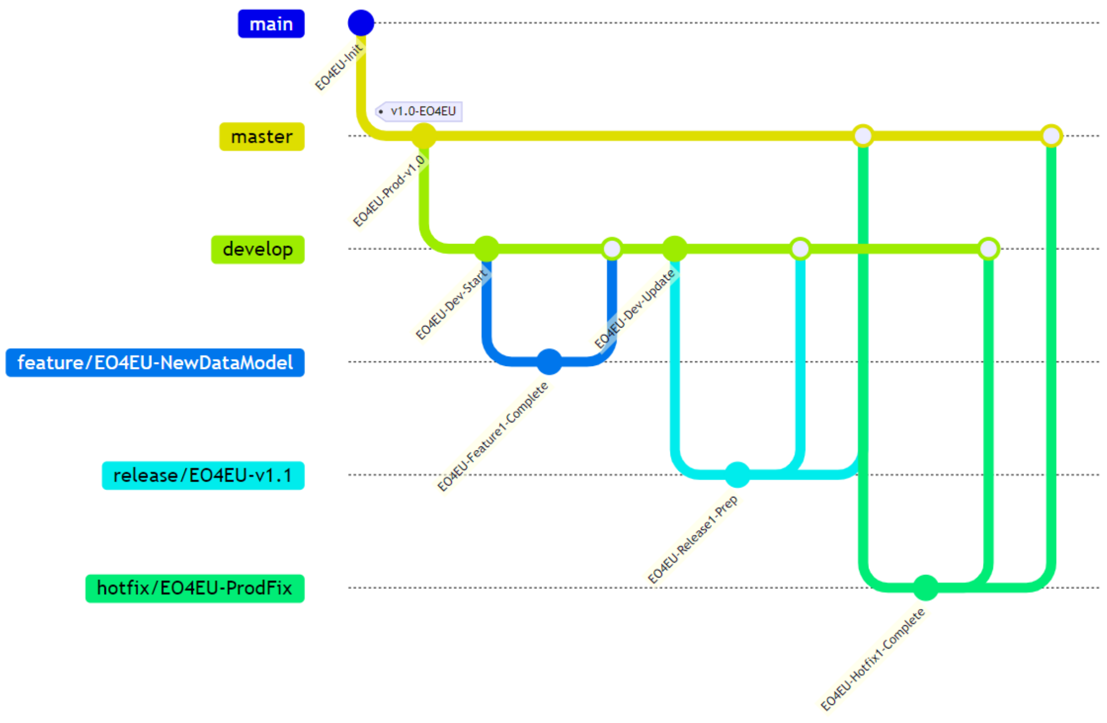


Figure 11 - GitFlow example

The Figure 11 displays the Git diagram of a generic EO4EU GitFlow process.The description of the steps followed is presented below:

1. Project Initialization (**EO4EU-Init**): The GitFlow workflow begins with the initialization of the EO4EU project. This initial commit sets up the project's repository.
2. Master Branch: The master branch is created following the initial commit. This branch represents the production environment and will contain the production-ready code.
3. Release in Production (**EO4EU-Prod-v1.0**): A commit tagged ***v1.0-EO4EU*** is made on the master branch, representing the first official production release of the EO4EU project.

4. Development Branch: A develop branch is branched out from the master branch. This branch is used for ongoing development and integration of new features.
5. Starting Development (**EO4EU-Dev-Start**): Development begins on the develop branch with the first commit, marking the start of new feature development.
6. Feature Branch (**feature/EO4EU-NewDataModel**): For the development of a new data model, a feature branch is created from the develop branch. This allows for isolated development of the new feature.
7. Completing Feature Development (**EO4EU-Feature1-Complete**): Once the new data model feature is developed and tested, it's committed to the feature branch.
8. Merging Feature into Develop: The completed feature branch is then merged back into the develop branch, integrating the new data model into the main development line.
9. Preparing for Next Release (**release/EO4EU-v1.1**): A release branch is created from develop to prepare for the next version (v1.1) of the EO4EU project. This branch is for final adjustments and bug fixes before the production release.
10. Release Preparations (**EO4EU-Release1-Prep**): The release branch is updated with final preparations and bug fixes for the upcoming version.
11. Merging Release into Master and Develop: After finalization, the release branch is merged into both master and develop, ensuring that the new version is deployed to production and that the development branch is updated.
12. Hotfix Branch (**hotfix/EO4EU-ProdFix**): If critical issues are found in the production version, a hotfix branch is created from the master branch to provide quick fixes.
13. Completing and Merging Hotfix (**EO4EU-Hotfix1-Complete**): Once the hotfix is complete, it's committed to the hotfix branch and then merged back into both master and develop to ensure the issue is resolved in both production and ongoing development.

This workflow represents a structured approach to managing development, features, releases, and hotfixes, ensuring stability and continuous integration in the EO4EU project.

By incorporating GitFlow into EO4EU, we achieve an improvement in managing software lifecycles, from planning to production. This workflow model promotes orderly management of dependencies between various software components and effectively supports the management of multiple environments such as development, test, and production. Furthermore, it fosters smooth and organized collaboration between teams, which is vital in an interdisciplinary project like EO4EU.

In conclusion, the integration of GitFlow in EO4EU represents a natural evolution of the software development process, aligning perfectly with the use of GitLab and GitLab CI/CD already in place in the project. This approach not only improves the quality and efficiency of software development but also helps creating a solid foundation for the future development and expansion of the EO4EU project.

3.8 EO4EU Container Registry

The CI/CD pipelines seen in the previous sections are used to automate the application building and deployment process. The images built by these pipelines need to be stored in an archive, namely a container registry.
This approach allows the EO4EU teams to efficiently manage Docker images, ensuring that they are always available and updated for use in the various components of the project. Also, the container registry makes it easy to share Docker images among team members and provides granular control over permissions and access to images.
Using a Docker Registry helps to simplify the image sharing with other team members or the community. Images can be accessed from any location via a URL. It helps also to keep track of different versions of the Docker images, making it easy to roll back or restore a specific version in case of problems. Moreover, the container registry can be configured with access policies and permissions,

ensuring precise control over image access. Finally sharing images can be used by multiple projects, reducing duplication of resources.
GitLab integrates a container registry functionality in the "Container Repository" section (Figure 12). This feature allows to store and manage Docker images directly within GitLab. In this way, being already within the GitLab ecosystem, it is not necessary to use external services to archive Docker images. Docker images stored in the Artifact section of GitLab are easily accessible from associated GitLab projects, simplifying resource management. Additionally, GitLab offers version control tools to track changes to Docker images.

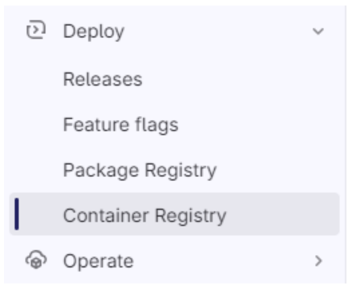
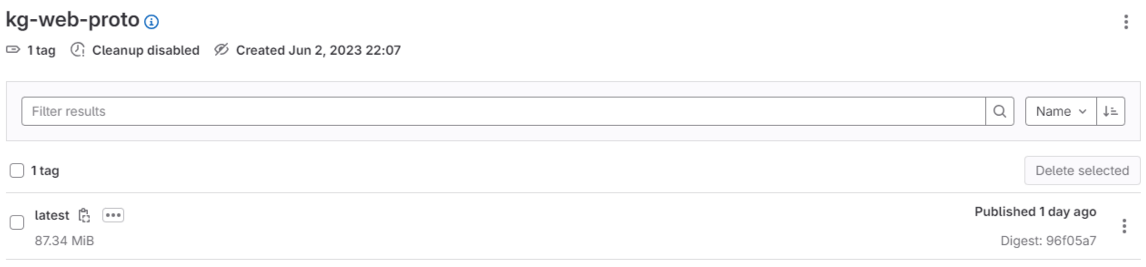


Figure 12. Container Registry section

EO4EU integrates the "Container Registry" section into all its GitLab projects. Gitlab "Container Registry" also provides versioning feature as can be seen in the image list of the knowledge graph shown in Figure 13, where the "latest" image tag is present.

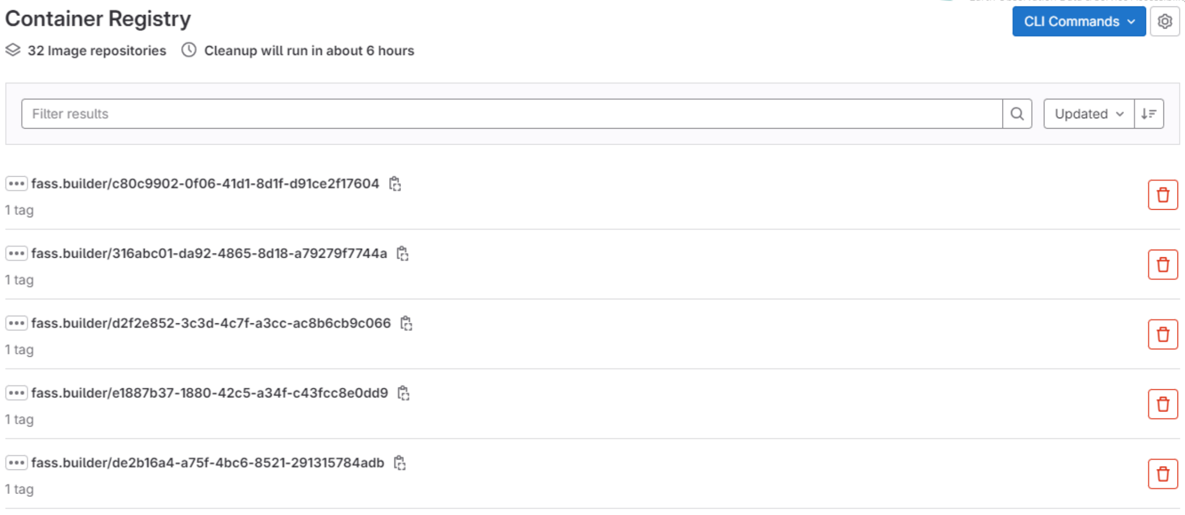

kg-web-proto

1 tag   Cleanup disabled   Created Jun 2, 2023 22:07

latest
87.34 MiB
Published 1 day ago
Digest: 96f05a7

Figure 13. Knowledge Graph component container registry

In other cases, the use of tagging activity is more intensive such as for "open-faas" components, where tags were used to provide a unique reference to the generated Docker images (Figure 14).


Container Registry

32 Image repositories · Cleanup will run in about 6 hours

fass.builder/c80c9902-0f06-41d1-8d1f-d91ce2f17604
1 tag

fass.builder/316abc01-da92-4865-8d18-a79279f7744a
1 tag

fass.builder/d2f2e852-3c3d-4c7f-a3cc-ac8b6cb9c066
1 tag

fass.builder/e1887b37-1880-42c5-a34f-c43fcc8e0dd9
1 tag

fass.builder/de2b16a4-a75f-4bc6-8521-291315784adb
1 tag

Figure 14. Openfaas container registry

After the storage of the Docker image to the GitLab container registry, deployment operations can start. In the upcoming project phase, newly developed CI/CD pipelines will automate deployment processes, enabling containers to run on the Kubernetes environment upon committing changes to the GitLab repository.

4 Integration and Validation Tests

In this chapter, the employed environments such as Test, Staging and Production are described. Each environment plays a crucial role in the software development and release process, enabling thorough testing and efficient implementation of new features and improvements.

Details of the integration tests and validation processes are also exposed, highlighting how these tests are fundamental to maintaining the high quality and reliability of the system. The approach taken to conducting unit testing and quality control is accurately illustrated, providing a clear view of the strategies adopted to ensure that each component of the EO4EU system functions interacts effectively with the other elements of the project.
The integration and validation tests are essential for the continuous improvement of the EO4EU platform. They ensure that the tools and practices implemented in our CI/CD processes (Chapter 2) and the user interface functionalities (Chapter 5) meet the highest standards of quality and reliability.

4.1 Environments

The entire infrastructure includes the full implementation both at the infrastructure level and at the organization level of three environments (Test, Staging and Production).
Each environment plays a specific and fundamental role in the software development lifecycle, facilitating a smooth and controlled transition from development to final product release. This structure allows to efficiently manage the various stages of development, testing, and release, ensuring that each release is solid, secure, and ready for use by end users. The clear distinction and effective management of these environments are essential to maintain the integrity and quality of the software produced within the EO4EU project.

4.1.1 *Test Environment*

It is a controlled test environment. The primary purpose is design-level testing and evaluation to assess the functionality, performance, and reliability of platform in base component level. At this level, unit tests for each component ensure expected operation regardless of the correct operation of the parts of the platform.

4.1.2 *Staging Environment*

It acts as a simulation of the production platform where end-to-end scenarios are validated before deployment in production to identify and remediate potential problems. The staging environment mimics the production environment to ensure realistic conditions. Testing in this environment helps verifying the compatibility, security, and overall robustness of the platform. This environment is also used for testing and integration of large-scale components of the platform.

4.1.3 *Production Environment*

The production environment is the live and operational environment where the platform is released for end users. Unlike development or staging environments, the production environment is optimized for performance, scalability, and security. Deployment in the production environment is a critical phase that takes place when the tests in the staging environment are completed. The use of Kubernetes clusters in the entire platform enables the hot-swap deployment method, to minimize downtime.

4.2 Validation Process

In the EO4EU platform all components are categorized based on various characteristics. Initially, a first separation can be made at the level of interaction with the user. Based on this categorization, the components are divided into backend services (Kafka, Rest APIs etc) and Front-end Services. Another level of separation is the technologies that have been used. Multiple technologies can be integrated into one component. The validation process is divided into two stages. The first stage includes the unit tests which must be integrated and executed during the deployment process in the test environment and the second stage is the Quality Assurance in the staging environment. The unit tests concern the base components of the platform, while the second control level concerns the high-level components.

No	Component	Frontend	Backend Service
1	Authentication Identity Platform (Keycloak)	X	X
2	Dashboard	X	
2.1	Knowledge Graph Query Component	X	
2.2	Workflow Editor	X	
2.3	Visualization	X	
2.4	S3 Bucket Explorer	X	
2.5	AR/XR	X	
3	Dashboards API		X
4	CLI	X	X
5	DSL Service		X
5.1	DSL Facade		X
5.2	DSL Engine		X

6	Fusion		X
7	FaaS	X	X
8	ML		X

Table 1: Component layer matrix

No	**Component**	Python	Java/ Kotlin	Typescript /Javascript	Docker	Unity	C#(.NET)
1	Authentication Identity Platform (Keycloak)		**X**		**X**		
2	Dashboard			**X**	**X**		
2.1	Knowledge Graph Query Component	**X**			**X**		**X**
2.2	Workflow Editor			**X**	**X**		
2.3	Visualization			**X**	**X**		
2.4	S3 Bucket Explorer			**X**			
2.5	AR/XR				**X**	**X**	
3	Dashboards API				**X**		**X**
4	CLI			**X**			**X**
5	DSL Service		**X**	**X**	**X**		
5.1	DSL Facade		**X**	**X**	**X**		
5.2	DSL Engine			**X**	**X**		
6	Fusion	**X**			**X**		
7	FaaS	**X**	**X**	**X**	**X**		
8	ML	**X**			**X**		

Table 2: Technologies matrix

4.2.1 Unit Tests

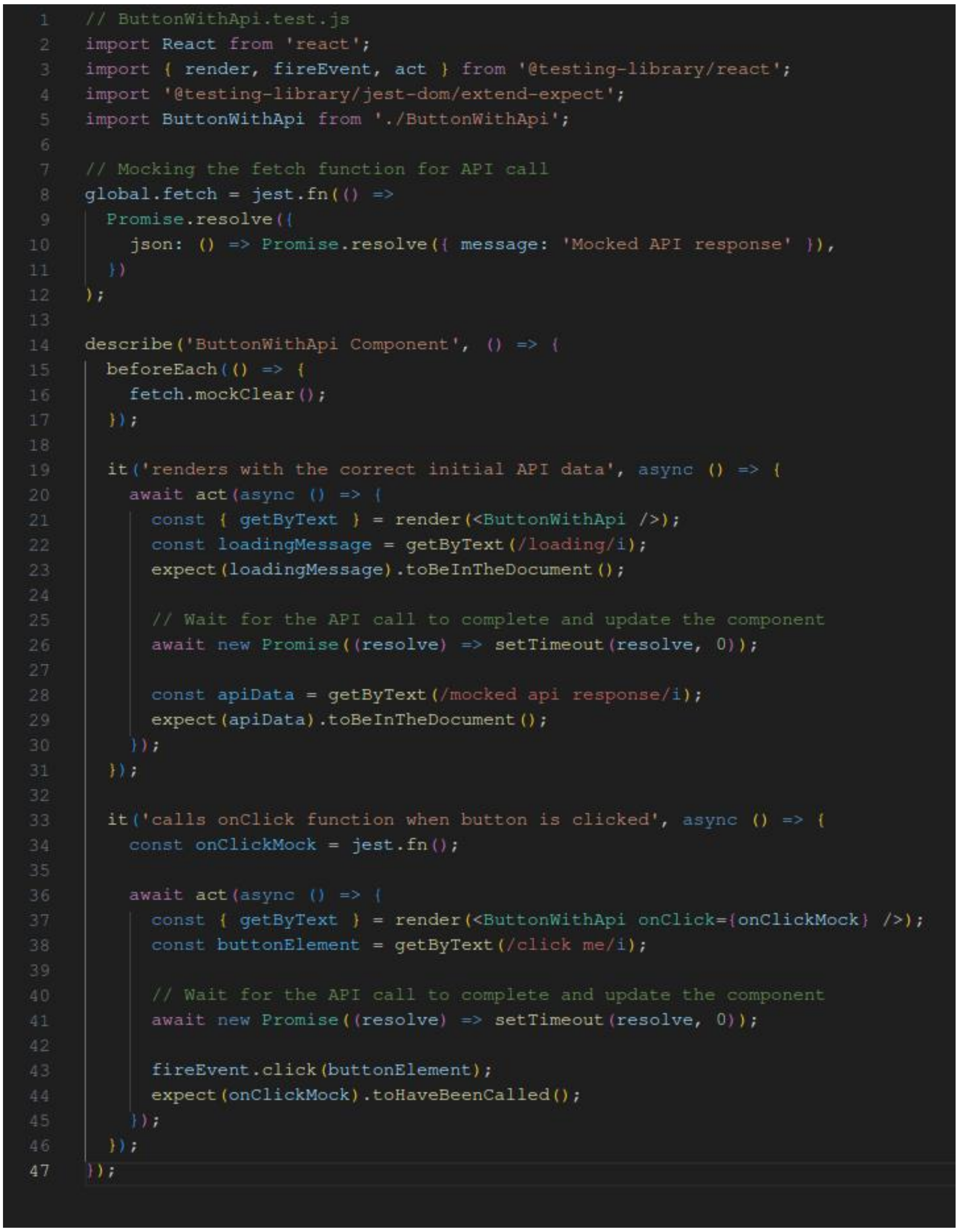
Testing procedures should be implemented based on the development technology. Processes are automatically executed within CI/CD pipelines. For the need to ensure the quality of controls, the tools that can be used at the component level were selected based on best practices.

Technology	**Test Frameworks**	**Testing Tools**
Java / Kotlin	JUnit, TestNG	Mockito, PowerMock
Python	PyTest, unittest	MagicMock, coverage.py
JavaScript / Typescript	Jest, Mocha	Sinon
C# (.Net)	NUnit, xUnit	Moq, dotCover
PHP	PHPUnit	Prophecy, Codeception
Go	testing	

Docker	Goss, Bats	Container - Structure Test, Dive

Table 3: Unit Test Matrix for Different Technologies

As an example, the test of a functional feature implemented with Typescript technology and the React framework through an automated process can be presented (Figure 15).



```
// ButtonWithApi.test.js
import React from 'react';
import { render, fireEvent, act } from '@testing-library/react';
import '@testing-library/jest-dom/extend-expect';
import ButtonWithApi from './ButtonWithApi';

// Mocking the fetch function for API call
global.fetch = jest.fn(() =>
  Promise.resolve({
    json: () => Promise.resolve({ message: 'Mocked API response' }),
  })
);

describe('ButtonWithApi Component', () => {
  beforeEach(() => {
    fetch.mockClear();
  });

  it('renders with the correct initial API data', async () => {
    await act(async () => {
      const { getByText } = render(<ButtonWithApi />);
      const loadingMessage = getByText(/loading/i);
      expect(loadingMessage).toBeInTheDocument();

      // Wait for the API call to complete and update the component
      await new Promise((resolve) => setTimeout(resolve, 0));

      const apiData = getByText(/mocked api response/i);
      expect(apiData).toBeInTheDocument();
    });
  });

  it('calls onClick function when button is clicked', async () => {
    const onClickMock = jest.fn();

    await act(async () => {
      const { getByText } = render(<ButtonWithApi onClick={onClickMock} />);
      const buttonElement = getByText(/click me/i);

      // Wait for the API call to complete and update the component
      await new Promise((resolve) => setTimeout(resolve, 0));

      fireEvent.click(buttonElement);
      expect(onClickMock).toHaveBeenCalled();
    });
  });
});
```

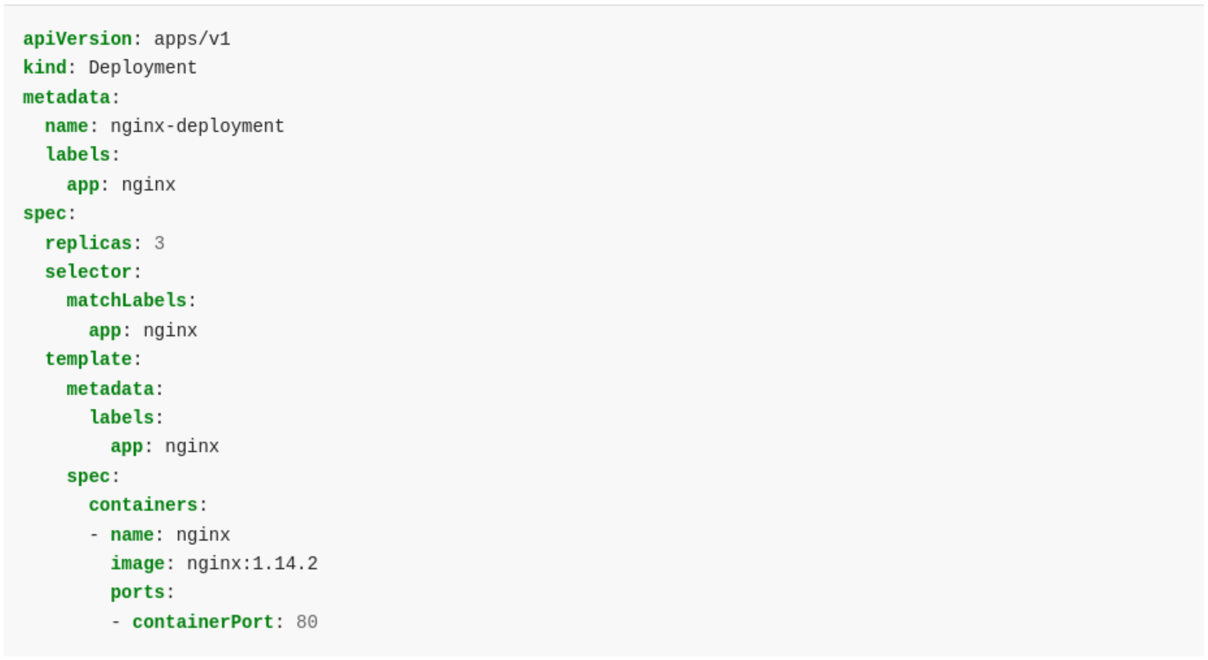
Figure 15. Jest emulates an API Call and the describe tool emulates the behaviour of click event.

4.2.2 Quality Assurance

Quality Assurance (QA) for the EO4EU platform involves procedures to ensure the reliability, performance, and security of the whole platform. The testing process typically begins with the design of end-to-end scenarios that includes all functional and non-functional parts.

4.3 Deployment Process (Release to Production)

After the completion of the tests in the staging environment, all the components can be transferred to the production environment. A "hot swap" deployment in Kubernetes typically involves updating an application or service with minimal downtime by rolling out the new version while the previous one is still running. To achieve the goal, each component must include a Kubernetes Deployment YAML file with the production settings (Figure 16). A central pipeline with access to all platform repositories can be deployed with the help of Gitlab Agent to all structural components.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Figure 16. Basic deployment configuration file for Kubernetes Platform

5 User guide and Graphical User Interface

This section describes the user functionality currently available in the EO4EU platform. The section and the functionalities whose content has not yet been completed will therefore be described in more detail in the final version of the deliverable (D4.8) at M33.

In the EO4EU platform the user dashboard is the central point of the user interface, designed to offer intuitive and secure access to data analyses and graphical visualizations. It is made up of several software components that work in synergy to optimize the user experience.

24Specific instructions on how to use each of these components and functions will be provided in subsequent sections of the user manual.

5.1 Authentication

Funded by
the European Union

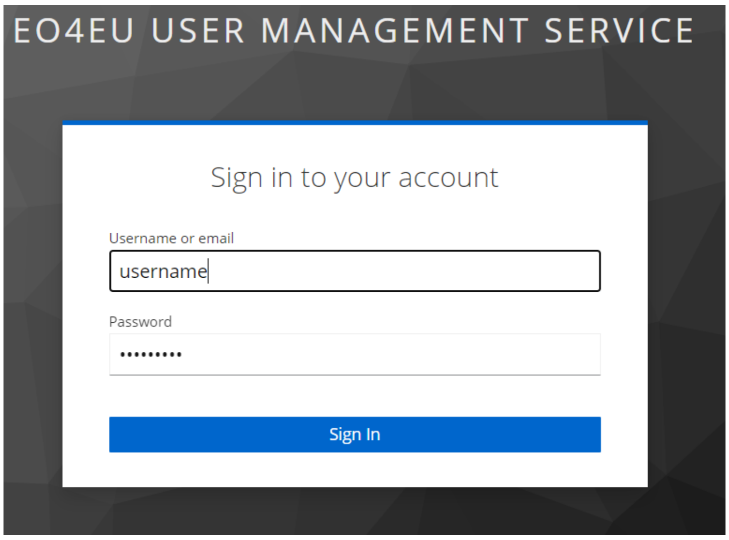
The robust authentication system adopted by EO4EU ensures that only authorized users can access the platform.

To authenticate and therefore access the EO4EU platform, a user must first have the appropriate access credentials (username and password). These credentials must be requested from the EO4EU system administrator.

Once the credentials have been obtained, to access the platform it is necessary to activate the authentication process using an OTP (One-Time Password) code. The OTP code is a disposable password, valid only for a single access session or transaction, which guarantees high security standards and solves the problems associated with the use of the traditional password.

Authentication instructions:

1. After the user tries to access the dashboard at the following link: https://dashboard.dev.wekeo.apps.eo4eu.eu/home
2. The system redirects the navigation to: https://auth.apps.eo4eu.eu/auth/realms/EO4EU/protocol/openid-connect/auth
3. The user fills in the login form with username and password (Figure 17).


EO4EU USER MANAGEMENT SERVICE

Sign in to your account

Username or email: username

Password: •••••••••

Figure 17. Login

4. The system will ask to use an OTP as shown in Figure 18 (such as Google Authenticator).

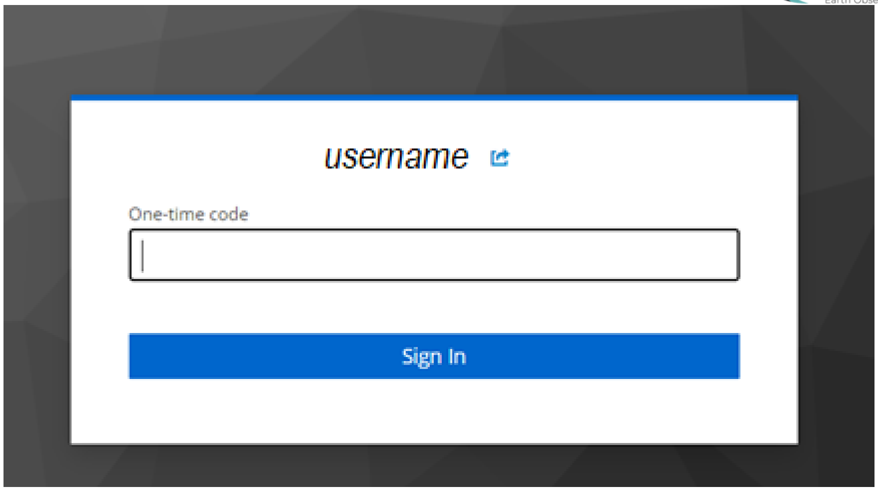


Figure 18. EO4EU OTP form

After completing the authentication process the system redirects the navigation to the EO4EU dashboard, user can start using the platform.

5.2 Dashboard Navigation

As stated above, the dashboard is the central access point to the EO4EU user functionalities. After the log in, the user lands in the home page of the dashboard.
The Home screen shows a menu on the left side. Each menu item represents a feature. The related features will open in the right part of the window. The EO4EU Front-end web-interface provides the following applications to the end-user:

- Home
- Dashboard
- Data Visualization
- Console
- Preferences
- History
- Account
- Sign-Out

These application are available through the main menu as shown in Figure 19.

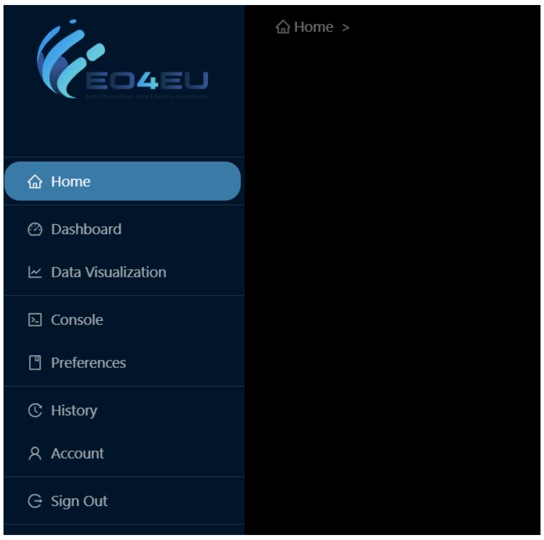


Figure 19. EO4EU Platform Front-end Web Interface

The Dashboard software component is designed and developed as a bundle of software instances and will activate all the related possible applications-services as shown below in Figure 20, in the right part of the window, such as: Knowledge Graph (KG), Workflow Editor, Elastic Search, S3 Bucket General, AI/ML Marketplace, Web XR/VR, and Global Store Records. Each software instance has its own unique operations and characteristic functionalities.

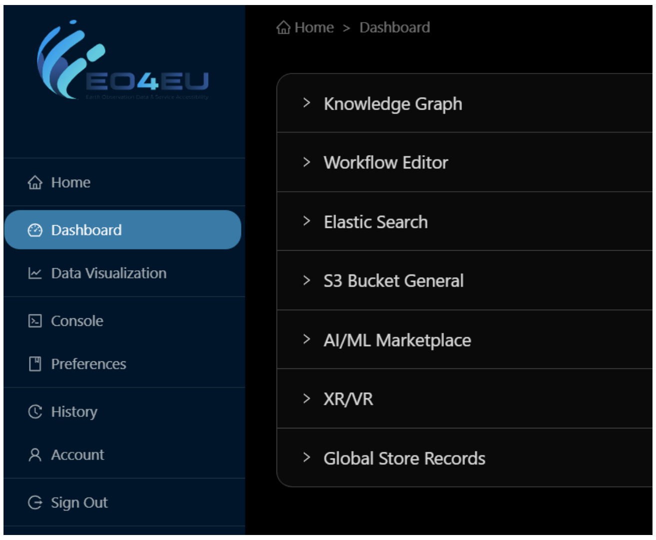
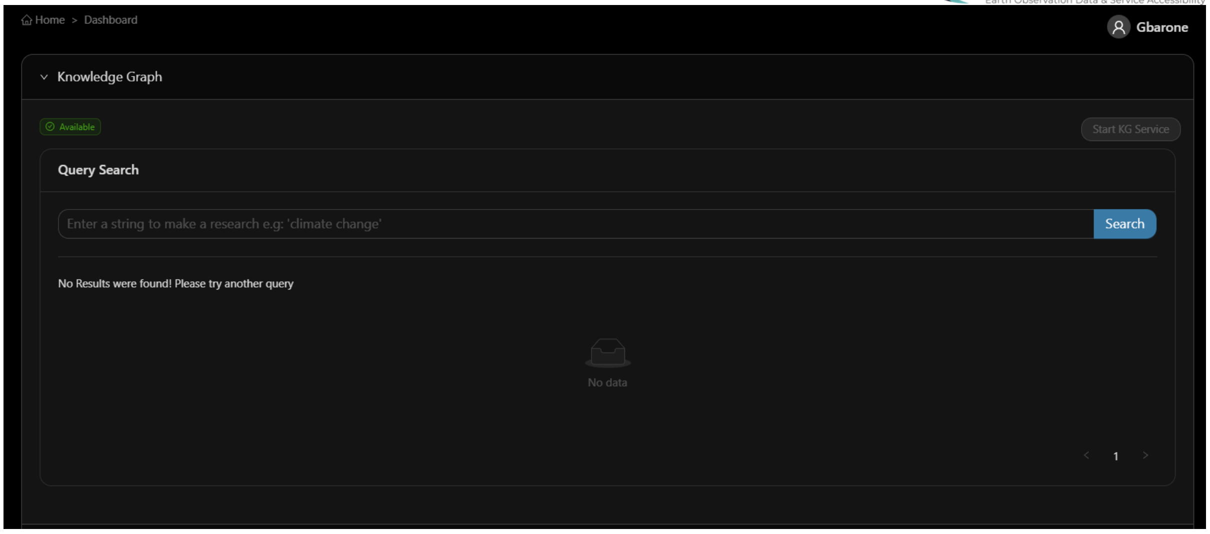


Figure 20. Dashboard – Software Instances-Applications-Service

Dataset Download

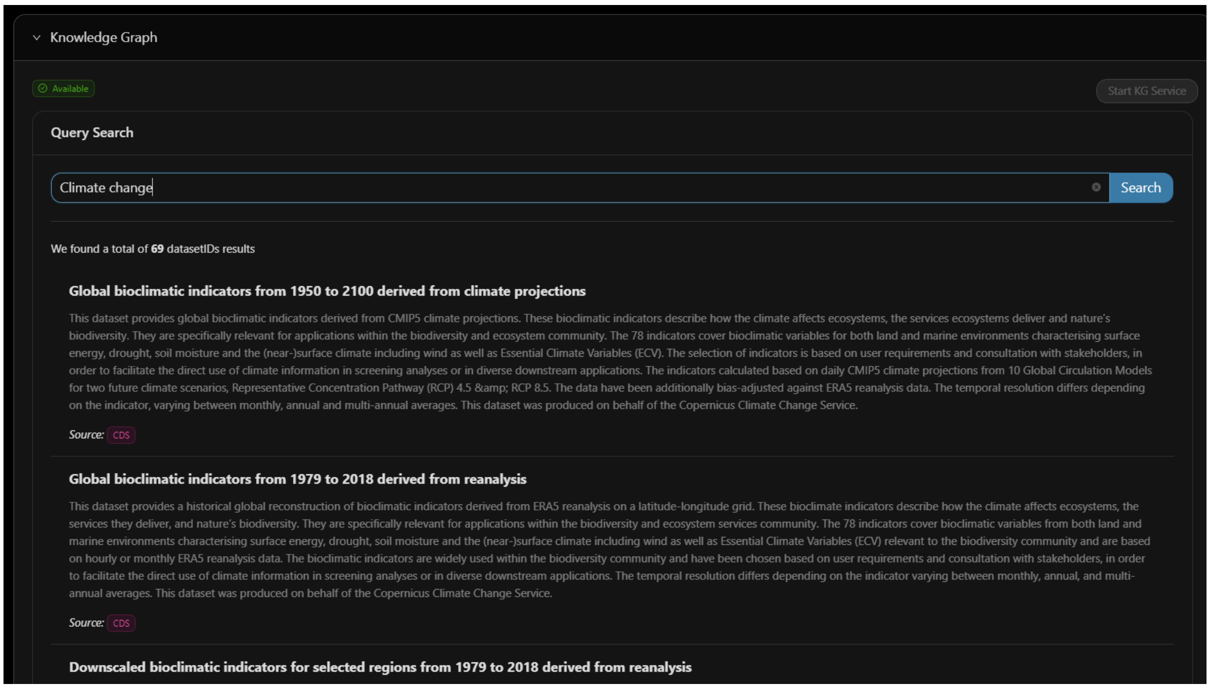
The EO4EU knowledge graph enables users to access and explore a plethora of Earth Observation (EO) data and derive valuable insights. The KG integrates disparate datasets so that users can explore interconnected data points. Through its semantic search capabilities, the knowledge graph enables users to locate specific information effortlessly using natural language queries, enhancing the overall search experience by grasping the context and relationships within the data (Figure 21).



No Results were found! Please try another query

Figure 21. Dashboard- KG Data Query Search

Users can explore the capabilities of the knowledge graph by entering specific keywords, like "climate change," into the "Query Search" section. Once entered, the system will generate results, as shown in Figure 22, in this example the research gets 69 relevant datasets. The results are displayed with titles and concise descriptions. By clicking on a dataset title, users can access detailed information, including a comprehensive description, variables, file format, data type, and a JSON representation of the dataset (Figure 23). This interactive functionality empowers users to explore the details of each dataset, improving their capacity to efficiently discover and comprehend relevant earth observation data.



We found a total of **69** datasetIDs results

Global bioclimatic indicators from 1950 to 2100 derived from climate projections

This dataset provides global bioclimatic indicators derived from CMIP5 climate projections. These bioclimatic indicators describe how the climate affects ecosystems, the services ecosystems deliver and nature's biodiversity. They are specifically relevant for applications within the biodiversity and ecosystem community. The 78 indicators cover bioclimatic variables for both land and marine environments characterising surface energy, drought, soil moisture and the (near-)surface climate including wind as well as Essential Climate Variables (ECV). The selection of indicators is based on user requirements and consultation with stakeholders, in order to facilitate the direct use of climate information in screening analyses or in diverse downstream applications. The indicators calculated based on daily CMIP5 climate projections from 10 Global Circulation Models for two future climate scenarios, Representative Concentration Pathway (RCP) 4.5 &amp; RCP 8.5. The data have been additionally bias-adjusted against ERA5 reanalysis data. The temporal resolution differs depending on the indicator, varying between monthly, annual and multi-annual averages. This dataset was produced on behalf of the Copernicus Climate Change Service.

Source: CDS

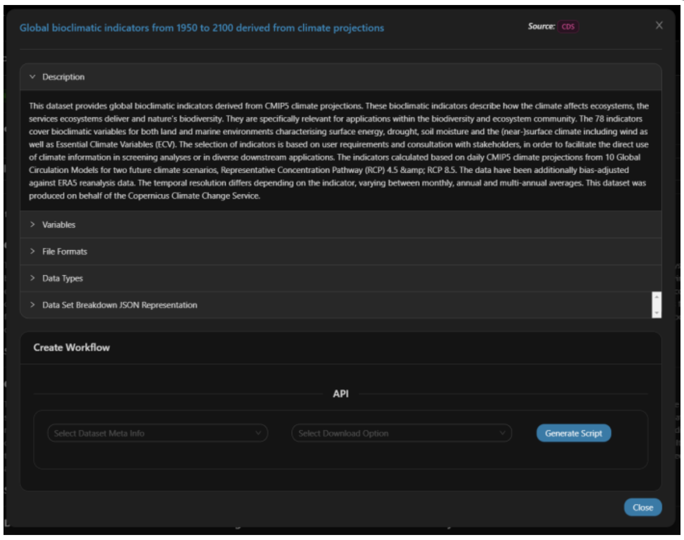
Global bioclimatic indicators from 1979 to 2018 derived from reanalysis

This dataset provides a historical global reconstruction of bioclimatic indicators derived from ERA5 reanalysis on a latitude-longitude grid. These bioclimate indicators describe how the climate affects ecosystems, the services they deliver, and nature's biodiversity. They are specifically relevant for applications within the biodiversity and ecosystem services community. The 78 indicators cover bioclimatic variables from both land and marine environments characterising surface energy, drought, soil moisture and the (near-)surface climate including wind as well as Essential Climate Variables (ECV) relevant to the biodiversity community and are based on hourly or monthly ERA5 reanalysis data. The bioclimatic indicators are widely used within the biodiversity community and have been chosen based on user requirements and consultation with stakeholders, in order to facilitate the direct use of climate information in screening analyses or in diverse downstream applications. The temporal resolution differs depending on the indicator varying between monthly, annual, and multi-annual averages. This dataset was produced on behalf of the Copernicus Climate Change Service.

Source: CDS

Downscaled bioclimatic indicators for selected regions from 1979 to 2018 derived from reanalysis

Figure 22. Data Sets Access based on Sorting Relevance Algorithm



Global bioclimatic indicators from 1950 to 2100 derived from climate projections

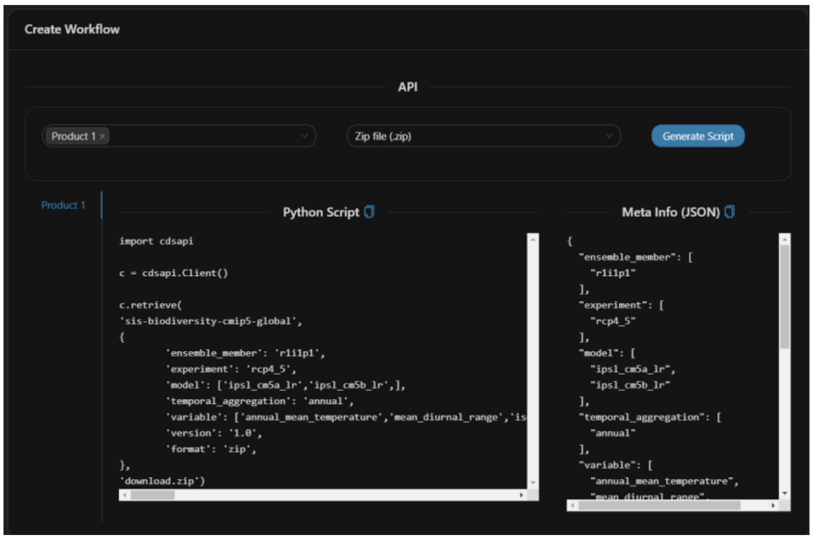
Source: CDS

Description

This dataset provides global bioclimatic indicators derived from CMIP5 climate projections. These bioclimatic indicators describe how the climate affects ecosystems, the services ecosystems deliver and nature's biodiversity. They are specifically relevant for applications within the biodiversity and ecosystem community. The 78 indicators cover bioclimatic variables for both land and marine environments characterising surface energy, drought, soil moisture and the (near-)surface climate including wind as well as Essential Climate Variables (ECV). The selection of indicators is based on user requirements and consultation with stakeholders, in order to facilitate the direct use of climate information in screening analyses or in diverse downstream applications. The indicators calculated based on daily CMIP5 climate projections from 10 Global Circulation Models for two future climate scenarios, Representative Concentration Pathway (RCP) 4.5 &amp; RCP 8.5. The data have been additionally bias-adjusted against ERA5 reanalysis data. The temporal resolution differs depending on the indicator, varying between monthly, annual and multi-annual averages. This dataset was produced on behalf of the Copernicus Climate Change Service.

Figure 23. Dashboard – Data Set Break Down based on pre-filtering mechanism

Selecting the dataset, the dashboard performs different APIs call to the Knowledge graph which executes smart pre-filtering mechanism and a sorting algorithm for the ten most relevant ID datasets, as result of this interaction in the subsection "Create Workflow" products and features become accessible for selection by the end-user. It is possible to choose the products available (one or more) for the specific dataset and the download file format and click on "Generate Script", which produces final MetaInfo and MetaData, including the essential Python script required for the Workflow Editor's pipeline processing. Multiple API calls are then made by the dashboard through the KG to obtain the Metainfo of the chosen ID dataset and its corresponding Python script in the Dashboard.
These details are subsequently sent to the Workflow Editor via the openEO API and also through a Kafka communication topic based on the user's selection. It's important to note that this automated mechanism is presently in development, serving communication and integration testing purposes. This automated mechanism is still under development as it serves communication and integration testing purposes for the current time, but it will be automated in a single API Call with a dynamic response to the Workflow Editor in Phase B Technical Development.
In the current version, user can select end copy the script and metainfo JSON file to use them in the creation of the workflow described in the next section (Figure 24).



```
import cdsapi

c = cdsapi.Client()

c.retrieve(
'sis-biodiversity-cmip5-global',
{
        'ensemble_member': 'r1i1p1',
        'experiment': 'rcp4_5',
        'model': ['ipsl_cm5a_lr','ipsl_cm5b_lr',],
        'temporal_aggregation': 'annual',
        'variable': ['annual_mean_temperature','mean_diurnal_range','is
        'version': '1.0',
        'format': 'zip',
},
'download.zip')
```

```
{
  "ensemble_member": [
    "r1i1p1"
  ],
  "experiment": [
    "rcp4_5"
  ],
  "model": [
    "ipsl_cm5a_lr",
    "ipsl_cm5b_lr"
  ],
  "temporal_aggregation": [
    "annual"
  ],
  "variable": [
    "annual_mean_temperature",
    "mean_diurnal_range",
```

Figure 24. Dashboard-KG MetaInfo and Python Script generation

5.3 Workflow creation and running

In the EO4EU project, a workflow is essentially a sequence of tasks organized systematically to process and analyse EO data.

The workflow is fundamental in the analysis of EO4EU data, the platform in fact allows to carry out predefined analyses and manage the data in the system, but also offers the possibility of carrying out customized analyses.

All this can be organized by the user through an interactive canvas, where each element is represented by a connectable block. By creating a pipeline, the user can easily configure the necessary settings, algorithms and scripts to perform the desired analysis.

The concept of workflow is pivotal as it defines how different components and tools interact to efficiently process EO data.

The process begins with the creation and design of the workflow. Users or system designers use the Workflow Editor (WFE) to specify and arrange tasks in a logical order. This workflow then undergoes a validation and compilation process through the Domain Specific Language (DSL) Engine, ensuring it adheres to the project's technical specifications. If the workflow is validated, it is compiled into a YAML format, making it ready for deployment.

Upon deployment, the workflow is executed within the EO4EU platform. This execution involves initiating the series of tasks or processes as defined. These steps include the ingestion, processing, analysis, and transformation of EO data, possibly using machine learning algorithms, data fusion techniques, or other methods. The aim is to streamline the transformation of raw EO data into actionable insights or useful information.

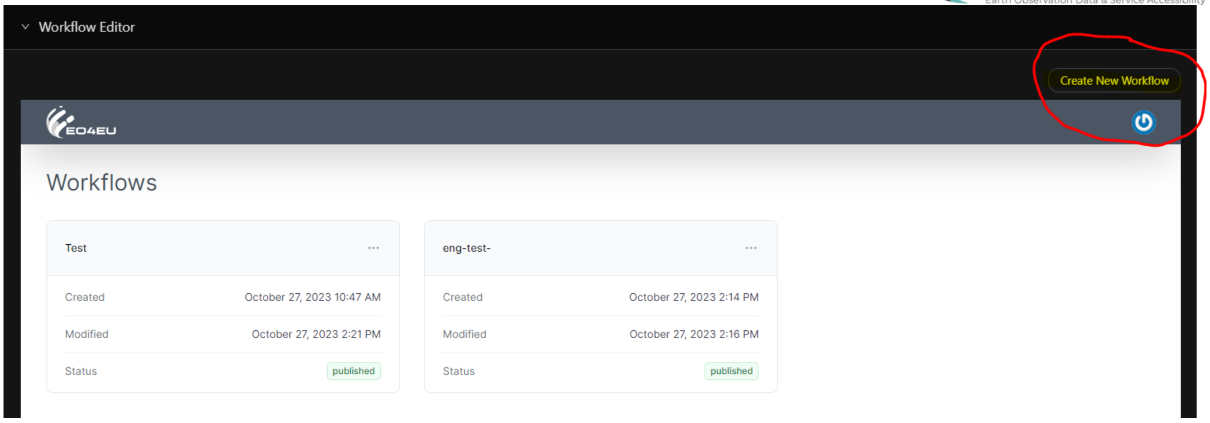
The flexibility and customization of these workflows allows users to meet specific data processing needs or objectives. There is a strong emphasis on automation to enhance efficiency and reduce manual intervention. Moreover, the integration of various technologies like AI, ML, cloud computing, and advanced data analytics is key to optimizing the processing and analysis of EO data.

An example of a typical workflow in EO4EU might start with retrieving satellite data from an EO database, followed by preprocessing tasks like cleaning and normalization. This data could then be passed through a Fusion Engine for correlation with other datasets and analysed using machine learning models from the AI/ML Marketplace. The workflow concludes with the generation of results such as predictive analytics, trend analysis, or visual maps, which are then used for decision-making.

Overall, workflows in EO4EU are crucial in transforming vast and complex EO data into practical, accessible, and actionable formats, unlocking the full potential of Earth observation for a variety of applications.

To start working with workflow, from the main menu select WorkFlow editor: The Workflow screen will open. Here the user can find all the previously created workflows. To open the workflow editor for an already existing workflow, the user can click on the menu icon and select “edit”(Figure 25).

Funded by
the European Union



Test	
Created	October 27, 2023 10:47 AM
Modified	October 27, 2023 2:21 PM
Status	published

eng-test-	
Created	October 27, 2023 2:14 PM
Modified	October 27, 2023 2:16 PM
Status	published

Figure 25. Workflow Editor

If a user wants to create a new workflow, "Generation scripts" are required. The primary purpose of these scripts is to fetch the actual EO data from the source selected.

To obtain a "Generation scripts", the user needs to follow the steps described in the previous section. Once the script is available, clicking on the "Create new Workflow" item will open the form for creating the new workflow and inserting the scripts, as shown in Figure 26.

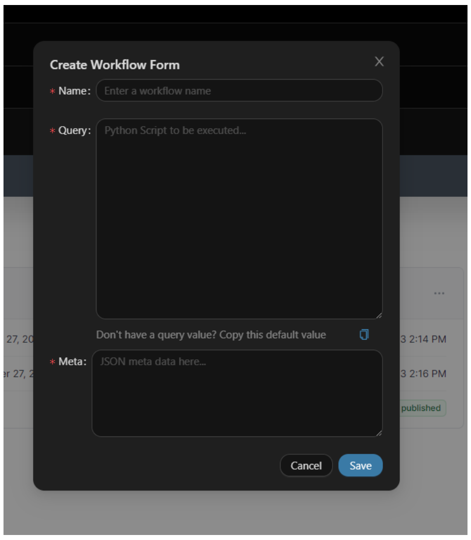

Create Workflow Form
* Name: Enter a workflow name
* Query: Python Script to be executed...
Don't have a query value? Copy this default value
* Meta: JSON meta data here...
Cancel
Save

Figure 26. Create New Workflow

Once the form is filled in with the script and the user clicks the "Save" button, the new workflow will appear in the list on the main screen (Figure 26).

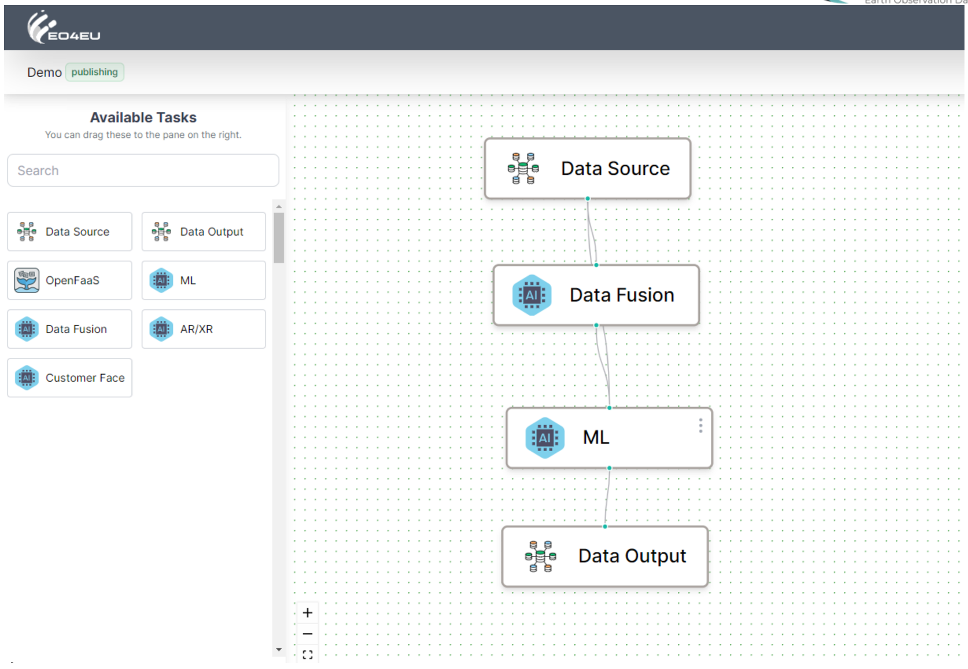


Figure 27 - Workflow Editor (WFE)

As shown in Figure 27, the system presents to the user a canvas with all the available tasks represented as blocks in the left column. The user can drag and drop the block on the central canvas and connect the block using the links (Figure 28).

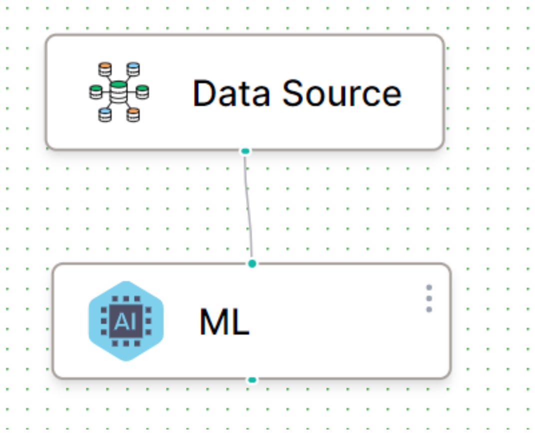
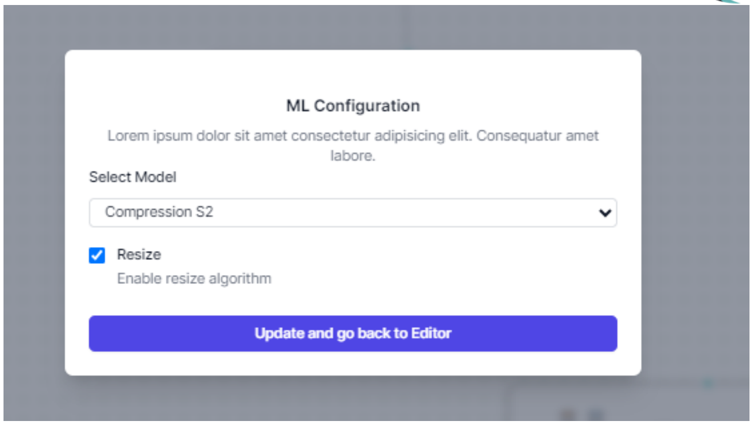


Figure 28 - Block connection

Blocks can also be configured by opening the configuration form available by clicking on the menu on the top right of the block (Figure 29).



ML Configuration

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur amet labore.

Select Model

Compression S2

[✓] Resize

Enable resize algorithm

Figure 29 - ML options

5.4 Data Visualization

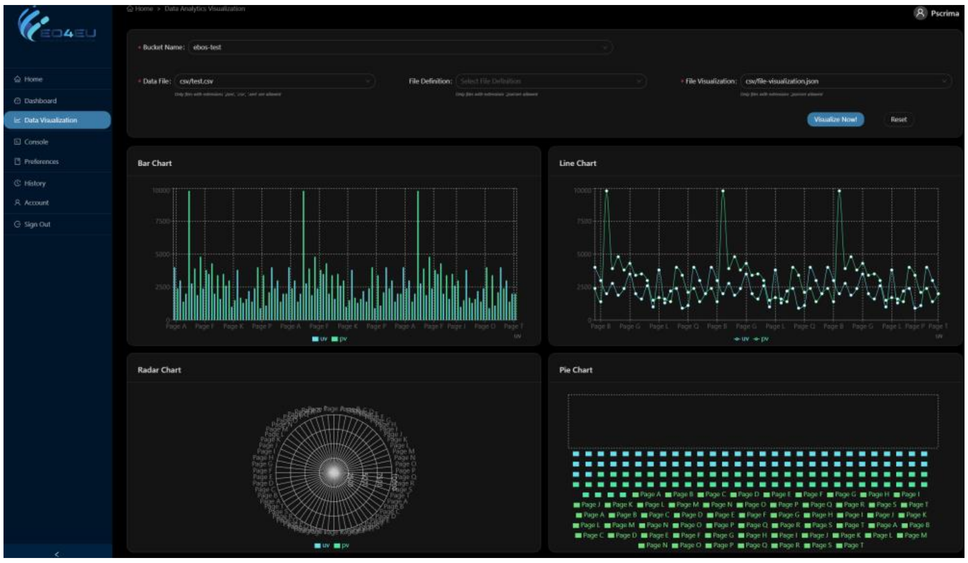
The data visualization functionalities in EO4EU are designed to convert raw data into meaningful visual insights, through dynamic charts and visual representations.

Data visualization takes place downstream of the dataset retrieval and workflow processing. The workflow process and knowledge graph need to generate the metadata necessary for data visualization. The data retrieved and processed in the EO4EU platform are then saved in the S3 storage.

The integration of the different components of EO4EU, and in particular of the data storage system, implies that the data retrieved and produced by the workflows are also visible from the data visualization component.

As described in deliverable D3.5, in the section related to the dashboard, aside from the data resulting from the processing, the workflow also produces the dataTypeConfig.json file, which defines the data type and structure-format of the metadata to be accessed and read. Another file produced by the workflow is dataVisConfig.json, which contains visualization settings, type of selected charts-plots, and data style to be plotted. These files provide the needed information for the data visualization to be automatically generated.

Currently, to elaborate and generate data visualizations, these files need to be selected manually. Also, at the time of writing the workflow does not directly generate these data, but there are some examples and demo data which allow the visualization of the graphs as shown in Figure 30.



Bucket Name: ebos-test

Data File: csv/test.csv

File Definition: Select File Definition

File Visualization: csv/file-visualization.json

Figure 30. Data Visualization

Another development that will be available in the next releases is the integration with Elasticsearch. Currently, the Data Visualization can provide 4 basic types of charts for simulating and visualizing the EO Data and EO Metadata generated by the KG and the WF Editor.

5.5 S3 Bucket

The S3 Bucket, used in the EO4EU project, is an object storage system, a modern and advanced storage technology. Unlike traditional file or block-based storage methods, object storage manages data as distinct objects. Each object includes the data itself, a unique identifier, and metadata. While traditional storage systems organize files into folder hierarchies, object storage stores them in a flat namespace, making them easily accessible and scalable.

The advantages of this type of storage are notable: greater scalability, ease of access and integration with modern cloud-based applications. Additionally, object storage provides advanced metadata management capabilities, improving classification and retrieval. Thanks to the use of a flat namespace, objects can be easily accessible from different applications and services, facilitating interoperability.

In the context of EO4EU, the S3 Bucket plays a crucial role in the integration of the different components of the system. The data collected and processed by the various tools and services within the EO4EU ecosystem are stored in a S3 Bucket. This centralized approach to data management greatly facilitates interoperability between components.

For example, during the research phase, the collected data is stored in the S3 Bucket. This centralized storage then allows the workflow component to easily retrieve this data for processing and analysis. Once the workflow has processed the data, the results are saved back to the S3 Bucket. These results can then be easily accessible by other components, such as data visualization tools, which read the data to present it in an understandable and interactive visual format.

Currently through the EO4EU dashboard, in the S3 Bucket section, it is possible to access the data processed (actually, retrieved and processed) during the operations carried out by the user. As can be seen in Figure 31, the interface shows the files which EO4EU processed to evaluate the request of the user. The metadata of the files can be explored, and the files can be downloaded as well.

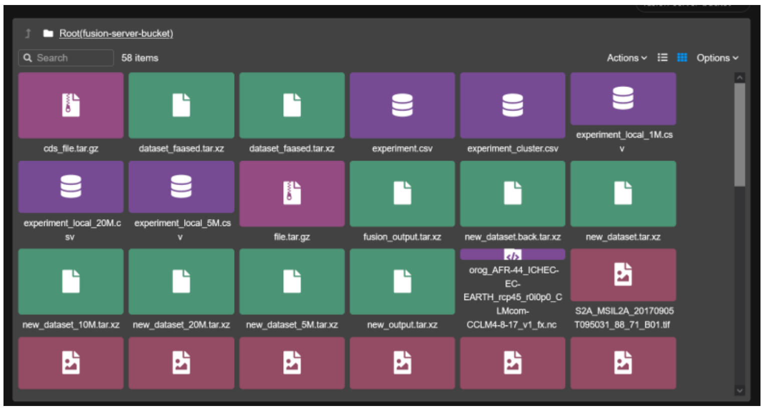


Figure 31 - S3 Bucket

5.6 XR/VR

The XR/VR component of the EO4EU project is a key aspect of the Customer Facing Services (CFS) and is designed to provide a more immersive and interactive experience for users engaging with EO data. The XR component is a web-based application that enables the possibility of visualizing and exploring the Workflow Editor (WFE) output data in an immersive environment, taking advantage both of Virtual Reality (VR) and Augmented Reality (AR) visualization capabilities.

The XR application is integrated in the Dashboard and it currently provides the visualization of WFE TIFF format output data with Virtual Reality capabilities (Figure 32), while the enabling of the Augmented Reality capabilities is foreseen for the phase B of the technical development.
Opening the XR/VR tab in the Dashboard, the EO data are displayed on the 3D model of the portion of terrain related to their context. The user can move, zoom in and out the 3D model and, in case the device he/she is using can support Virtual Reality experiences, a button for enabling the Virtual Reality visualization is shown on the screen. Clicking the button, the user is redirected to a newly opened web page where the 3D model is displayed in VR mode.
More advanced possibilities of user interactions will be developed in the phase B of the technical development.

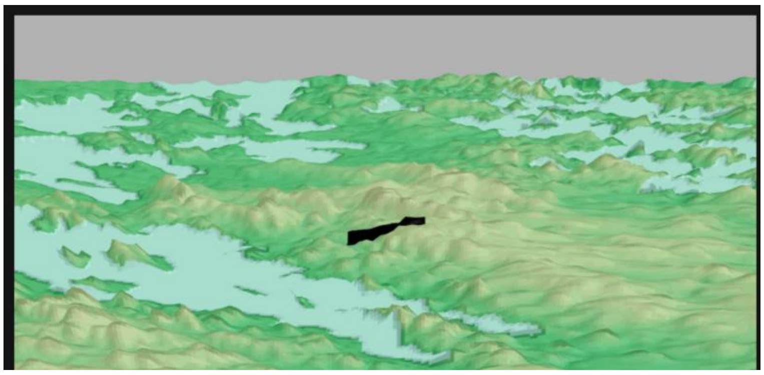


Figure 32 - VR/XR

6 Conclusions

This document reflects the initial phase of the EO4EU project, capturing the essential steps taken towards developing the EO4EU platform. It is clear how each component of the EO4EU platform, from the GitLab repository and CI/CD processes (Chapter 2) to our comprehensive testing methods (Chapter 4) and the user interface (Chapter 5), work cohesively to create a robust and user-friendly system for EO data analysis and management. While this first release is an important step forward, it represents the beginning of a journey towards achieving a robust and user-friendly platform for EO data analysis and management. The document acknowledges that while the current release lays the groundwork, there is an ongoing focus on enhancing the system's robustness and user-friendliness. These are key objectives for the final system, and current efforts are dedicated to iteratively improving the platform. This phase serves as a crucial foundation, informing future enhancements and setting the stage for the continuous evolution of the EO4EU ecosystem to meet the sophisticated needs of EO data utilization.