D4.5 - Release of -Serverless- Infrastructure Configuration management as a service (a)

Work Package	WP4, EO4EU Data Marketplace Ecosystem
Lead Author (Org)	Babis Andreou (NKUA)
Contributing Author(s) (Org)	Kakia Panagidi (NKUA), Nektarios Deligianakis (NKUA), Michalis Loukeris (NKUA), Vironas Korpas (NKUA)
Due Date	30.11.2023
Date	29.11.2023
Version	V2.0

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

Versioning and contribution history

Version	Date	Author	Notes
0.1	01.09.2023	Babis Andreou (NKUA)	TOC and V0.1
1.0	01.10.2023	ALL from NKUA	1st Version
1.1	01.11.2023	Rob Carrillo (Trust-IT)	1st Review
2.0	29.11.20123	ALL from NKUA	Final Version

Terminology

Terminology/Acronym	Description
CSA	Coordination and Support Action
DoA	Description of Action
EC	European Commission
GA	Grant Agreement to the project
FaaS	Function as a Service
PS	Provision Service

Table of Contents

List of Figures

List of Tables

NO TABLE OF FIGURES ENTRIES FOUND.

Executive Summary

This deliverable reports the progress made for Serverless Infrastructure in Task T3.2 of the WP3 during the first implementation cycle of the EO4EU project. The document provides an insight of the initial deployment and integration of the FaaS setup infrastructure presenting the architecture and the functionalities of the FaaS service that EO4EU platform will use. Finally, at end of this deliverable there are links of the repositories that are under development for the different components.

1. Introduction

1.1 Scope of D4.5

This deliverable describes deployment and integration of the Function as a Service (FaaS) setup infrastructure involved in the first implementation cycle. The document presents:

- The general technical approach
- FaaS hardware and software environment
- Foreseen refinements

1.2 Relation to other deliverables

A detailed description of the development status of the EO4EU platform was carried out in D2.2 “EO4EU End-user Requirements Analysis & Business process flows” and D2.4 “Technical, Operational and Interoperability specifications and Architecture”. The D2.2 includes the first version of user requirements of EO4EU and the baseline of all business processes, which will take place through a flowchart methodology. Furthermore, the D2.4 presents the system specifications for all different components of the EO4EU solution and a technical overview of the architecture of the proposed solution.

2. General Technical Approach

In the EO4EU platform, the main concern is to provide processing capabilities to the user. Capabilities that will extend the perspective of the user to the EO data. The Function as a Service (FaaS) functionality will provide a best effort hard coded processing that a user can adapt to fit the needs of his EO data processing. It is able to reserve the needed resources according to the work requirements and instantiate the needed functionality on demand . The outcome will be a FaaS system that is dynamic and versatile, processing a vast load of information with minimum delays.

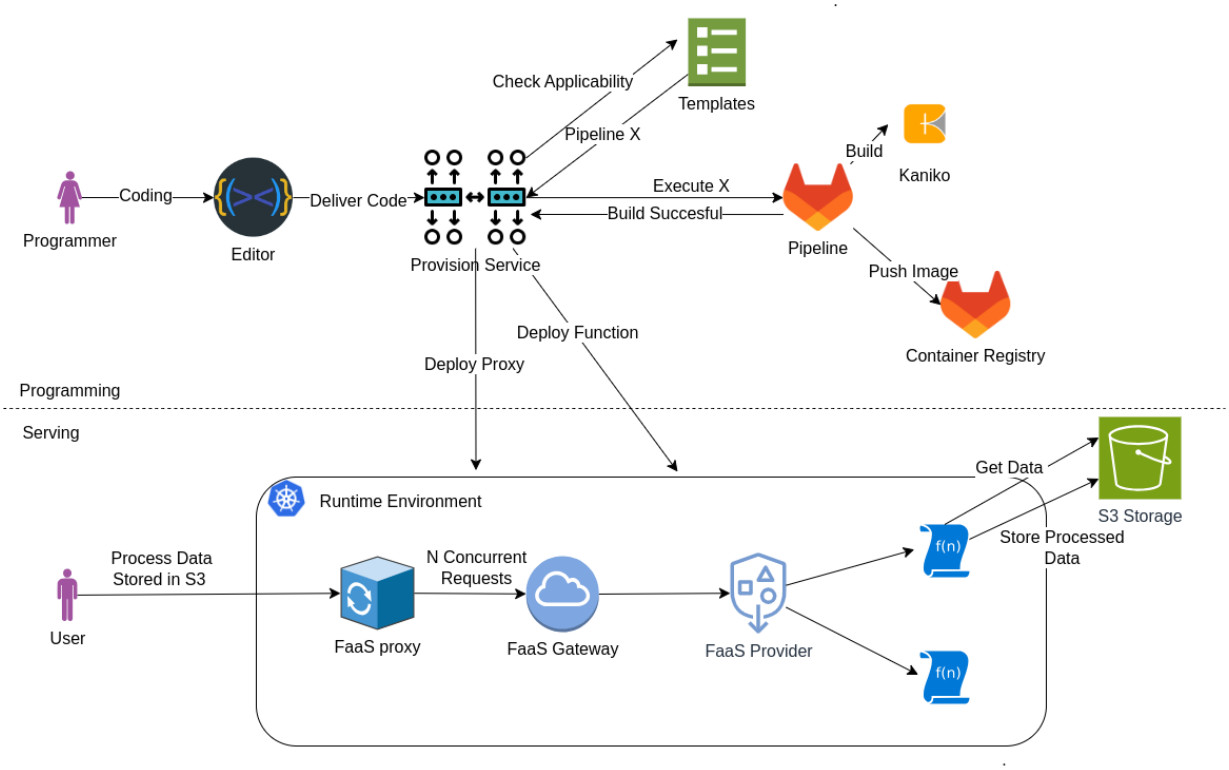

Figure 1 **– FaaS architecture**

EO4EU Kubernetes cluster will provide dynamic virtualisation environment based on the containerised Docker characteristics. The provided Kubernetes playground will be handled through an intuitive Provision Service component that conditionally can provide deployments to the cluster.

2.1 FaaS Proxy

2.1.1 Process decomposition

The nature of serverless functionality relies in lightweighted short time executions of a particular block of code (process). Trying to leverage FaaS service, the EO4EU platform avoids driving the enormous amount of data, that need to be processed, through one single instance of FaaS function and distributes the data load accordingly providing the capability to scale up or down, and discard the resources as required. Minimum intervention

When comes to data processing, interventionism in data is adding complexity that finally is ending up to false outcome. For this reason, the FaaS proxy is not intervening in any phase of the processing providing only an abstraction of the service.

2.1.2 Data distribution

Based in the templated form of the provided serverless function, the FaaS proxy delivers only the address of the data that is going to be processed within the FaaS function as reference cursor to the data storage. This templated form of the serverless function provides the capability to interact the data in the data store without prior knowledge of this store.

2.1.3 Provision Service

After the Provision Service (PS) receives confirmation and the pipeline successfully finished, it begins to create the Kubernetes deployment resource yaml files, to dispatch it to the cluster for creation and execution. The PS generates an appropriate name for this deployment which must be unique and follow the RFC1123 naming requirements, otherwise the Kubernetes instance of the cluster will reject it. Additional Kubernetes resource files have to be created, like a Kubernetes secret object, containing authorisation info of the container registry the FaaS Docker image resides. Failing to provide this will result in ImagePullBackOff errors inside the cluster.

2.2 Gitlab pipelines

The use of FaaS is inextricably linked to the use of Linux containers. At the same time, the high-risk process and execution of code, which the user enters to be executed on the platform, requires the use of automation tools for the production, of these Linux containerised applications.

Basic tools that offer the possibility of complete standardisation of the process are pipeline tools pronounced by DevOps platforms. In the current implementation, well-known tools such as Git Lab Runners, Jenkins Agents and Circle CI are evaluated. The Git Lab platform was chosen precisely because of its existence in the EO4EU project but also because it is provided as self-hosted. It can run on top of a k8s Cluster, and it is also having capacity for horizontal scaling.

For the prototyping part, the FaaS CLI scaffolding tool was used, which provides fully functional templates both for script-based programming languages such as Python, JavaScript, and for type of safe-object-oriented programming languages such as Java and C\#.

2.2.1 Pipeline description

The pipeline process is described entirely in a yaml file that includes all the processes. The basic structure consists of stages and individual jobs. The jobs that make up the basic structural unit are executed in independent runners and are isolated from each other.

Choosing to install runners on k8s provides an additional level of security and usability given that you run each task in a separate ephemeral deployment.

In addition, the containerised environment of the runner can use any image as base image with the possibility of selection both at the pipeline level and at the job level.

2.2.2 Jobs and Stages

The process of creating the production images for FaaS consists of two stages and a total of three jobs.

2.2.2.1 Build

The first stage named build contains two tasks that perform either the first purpose of encapsulating the user's input code in the appropriate template and the second the build docker image process.

The last stage in this particular architecture presented a set of difficulties related to the fact that the image builder required a privileged access level to the docker engine.

This situation was remedied by using Google's Kaniko tool, which give us the ability to produce docker images, inside a container (docker-in-docker architecture) without exposing structural elements of the operating system.

2.2.2.2 Deploy

The next and last stage is called deploy and it performs a task that aims to upload the image to the private docker repository of the EO4EU platform.

Dynamic configuration and on demand execution

Each pipeline supports the ability to define variables. These can either be defined by the environment (env variables) or defined within the pipeline. In our scenario, the pipeline requires as inputs the template that the user has chosen from the graphical interface (between Python, Javascript and Php) and the code that he wishes to execute. The group of templates that integrate python along with python editions for scientific purposes were built from the ground up. Action of strategic importance since the EO4EU platform

The PS forms the URL encoded request including all given specifications from the graphical user interface.

On the fly build

After the successful creation of the FaaS Docker image, the PS prepares the required deployment resource to submit it to the cluster. This creates the required pods which execute the FaaS inside the cluster.

2.3 The Code

You can find a full functional release of the serverless functionality in the link below

Credentials

User: review@eo4eu.eu

Password: tgc2eud8EMK5vpx_mnz

FaaS proxy:

https://gitpcomp.di.uoa.gr/eo4eu/wp3-data-orchestration-and-ml/t3.2-systems-and-services-orchestration/t3.2.1-systems-and-services/t3.2.1.3-orchestrator/t3.2.1.3.4-provision-manager/openfaas-functions

FaaS builder:

https://gitpcomp.di.uoa.gr/eo4eu/wp3-data-orchestration-and-ml/t3.2-systems-and-services-orchestration/t3.2.1-systems-and-services/t3.2.1.3-orchestrator/t3.2.1.3.4-provision-manager/openfass-builder

Platform link:

https://dashboard.dev.wekeo.apps.eo4eu.eu/dashboard

3. Conclusion

This document contains the full description of the components and functionalities of the Serverless infrastructure of the EO4EU platform. D4.5 provides the design and technical description of the

components but also a full functional release of the code that EO4EU development team creates for FaaS service.