Integrating and deploying heterogeneous components by means of a microservices architecture in the CROSSMINER project

Amin Boudeffa[1], Antonin Abherve[1], Alessandra Bagnato[1], Davide Di Ruscio[2], Márcio Mateus[3], Bruno Almeida[3]

[1]Softeam R&D Department, France

firstname.lastname@softeam.fr

[2]University of L'Aquila, Italy

firstname.lastname@univaq.it

[3]Unparallel Innovation, Lda, Portugal

firstname.lastname@unparallel.pt

Abstract. The CROSSMINER project is an open source project, which is motivated by the increasing dependence on existing open-source software (OSS) to develop new complex systems. The project is a follow-up of the previous OSSMETER project. The complexity and diversity of new CROSSMINER components and existing OSSMETER ones raised challenges related to the integration and the communications among heterogenous built-in components as well as addressing security aspects for the whole project. In this paper, we present a microservice architecture, which is implemented by relying on Docker to support the integration and deployment of the CROSSMINER components.

Keywords: Open Source Software, Heterogeneous Legacy Components, Integration, Deployment, Microservice.

Project data

- Acronym: CROSSMINER
- Title: Developer-Centric Knowledge Mining from Large Open-Source Software Repositories.
- Partners: The Open Group, University of York, University of L'Aquila, Athens University of Economics & Business, Bitergia, Castalia Solutions, Centrum Wiskunde & Informatica, Eclipse Foundation Europe, Edge Hill University, FrontEndART, OW2 Consortium, Softeam, The Open Group, University of L'Aquila, University of York, Unparallel Innovation.
- Start date: 1 January 2017.
- Duration: 36 months.
- Web site: https://www.crossminer.com/

1 Introduction

Open-source software (OSS) is computer software distributed with a license that allows access to its source code, free redistribution and the creation of derived works. Unlike commercial software which is typically developed within the context of the organization with a well-established business plan and commitment to the maintenance, documentation and support of the software, OSS is very often developed in a public, collaborative, and loosely-coordinated manner. This has several implications to the level of quality of OSS as well as to the level of support that OSS communities provide to users of the software they produce. Consequently, developing new software systems by reusing existing open source components raises challenges related to at least the following activities: i) searching for candidate components, ii) evaluating a set of retrieved candidate components to find the most suitable one, and iii) adapting the selected components to fit the specific requirements. Dependence on OSS projects can either be a blessing or a curse [1]. The ability to accurately assess the risks and benefits of adopting OSS projects as components is essential to the software development community at large.
The EU CROSSMINER project works toward providing a standard way to help the developer's community through increasing dependence on existing open-source software (OSS). The project aims to automatically extracting required knowledge by applying mining techniques of different information source of the OSS projects and then injecting them as recommendations into the developers' Integrated Development Environments (IDE), at the time they need it to make design decisions.
The paper is structured as follows: The next section provides a review of related works about the project. Section 3 gives the integration challenges of the heterogenous and legacy components. Section 4 presents the planned integration tasks of the CROSSMINER components. Section 5 describe the outline deployment approach of the CROSSMINER platform.

2 Related Work

The EU OSSMETER FP7[2] project developed a distributed and horizontally-scalable platform for incremental analysis of multiple dimensions of open-source software projects including their source code, communication channels, and bug tracking systems.
The aim of CROSSMINER is to extend the outcomes of the OSSMETER project and to deliver an integrated open-source platform that will support the development of complex software systems by (1) enabling monitoring, in-depth analysis and evidence-based selection of open source components, and (2) facilitating knowledge extraction from large opensource software repositories [1].
This paper is a follow up of the work already done on the EU CROSSMINER 2017 STAF-RPS [1] which will focus on the SOFTEAM's mission within the project as a consortium partner which oversaw the architecture specification, the legacy components integration and evolution.

3 The CROSSMINER's Components Complexity

3.1 Overview

The CROSSMINER project integrates a set of components, which built using completely a legacy technology stack. Furthermore, other components are added in the current project following the same architectural approach, which includes new source code and NLP (Natural Language Processing) measurement tools, system configuration DevOps tools, workflow-based knowledge extractors and advanced IDE (Integrated Development Environments).
Figure 1 presents the logical architecture of the CROSSMINER platform, by identifying the functional modules that compose the platform and identifying the contribution from the OSSMETER project.

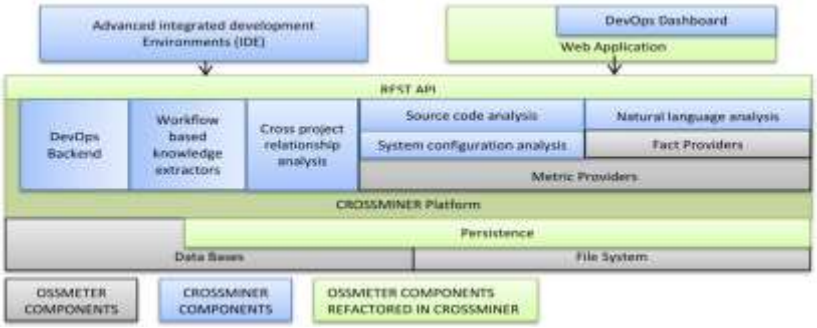

Figure 1. The CROSSMINER logical architecture.

These tools available on [3] were adapted and improved in the context of CROSSMINER activities and were included in the Metric Platform module, aggregating all the metric providers and metric execution infrastructures.

3.2 The integration challenges

The CROSSMINER project is a complex platform that assembles several components to provide an accurate analysis of open-source software. The complexity and diversity of these components including those legacy ones inherited from OSSMETER raised challenges related to the integration process to establish a common means of communication among the heterogenous built-in components. On the other side, the platform's frontend clients aim to have a seamless way to interact with these macro components through a unified entry-point rather than calling individual service endpoints that may help reduce and maintain the complexity level. Finally, by choosing a mean to connect these components implies the need to provide a flexible, secure, and efficient authentication and authorization scheme of the stack endpoints publicly exposed.
To handle all these issues, we explored some technologies inspired from microservice architecture to assembly those heterogenous components into a unified platform.

4 Integrating the Heterogeneous Legacy Components

Microservices are essentially independent software services that provide a specific business functionality in a software application. Each microservice is loosely coupled with other services. These services are self-contained and serve a single functionality.
The first integration task focuses on identifying the principal macro components, by delineating their perimeters and administering interactions between them, as defined in Figure 2. In this context, we have identified the macro components that support the services provided by the CROSSMINER platform and the clients composed of applications, which consume these services.

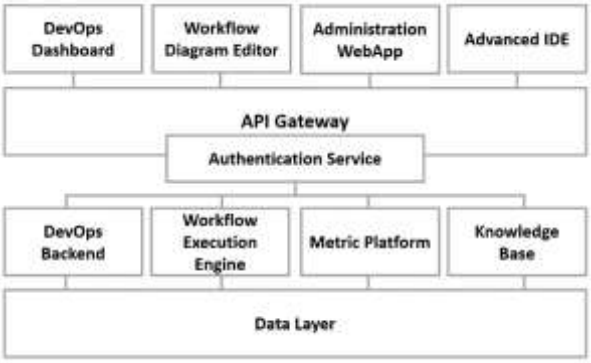

Figure 2. CROSSMINER macro-components architecture

In our integration approach, we have made the choice of not imposing a common technology for the implementation of the different services that compose the platform and did not make any assumptions about where and how CROSSMINER components will be deployed. Our only requirement is that the components must provide a REST endpoint for their services and that all communication between services and clients or between services itself must be implemented through such APIs.
The second integration aspect is to address the high-level complexity due to the communication between the front clients and the macro components. Here, the key concept is to provide a unified REST API that aggregates all the services provided by the other CROSSMINER components. The new component was added to the architecture: *API Gateway*. It is a pattern, which comes from microservices ecosystem and that represents a single point of control for frontend clients by allowing them to consume services provided by the different platform components (see Figure 2). Also, it acts as a reverse web-proxy that redirects client's requests to services provided by the platform components. With this approach the client only has to know the URL of the API Server to access all the services provided by the platform. At the same time, the REST API of the components can be refactored with no changes to the API provided by the API Gateway.
The last challenge is confronted with the need to secure access to the services provided by the CROSSMINER platform. To this end we setup an authentication system at the level of the API Gateway based on the JSON Web Token (JWT) security mechanism [5] as shown in Figure 2. The authentication service allows to secure the heterogeneous REST service bundle in a centralized way by allowing the CROSSMINER administration, when required, to protect specific resources and to define different authorization levels for different clients.

5 Deploying Heterogeneous Legacy Components

For the deployment of the CROSSMINER platform, as each of the CROSSMINER components has its own procedures and requirements for building and deployment, it is planned to follow a deployment strategy based on containers. This kind of approach provides several advantages related e.g., to the isolation of the components processes inside standalone containers, speed up the container's deployment, etc.
Regarding the container technologies, Docker has been considered as the favorite standard for this kind of technologies. This is due the fact the Docker execution engine as currently assuming a very stable behavior on the major operation systems (i.e. Windows, Mac OS, and several Linux distributions). Moreover, Docker images are supported by the majority of Cloud Platforms. Docker provides a tool for defining and running multi-container Docker applications – Compose. Compose uses a file to describe the services that compose the distributed application, by identifying aspects like: the images used by each service, the network configuration behind the services, the dependency between the services, the data volumes required by the distributed application, etc. Compose allows to create containers, configure them and the underlying network, and start the distributed application with a single command.
Figure 3 shows a diagram representing how the current CROSSMINER distributed application is structured.

Figure 3 – Diagram of CROSSMINER as a Docker distributed application

Currently, the application consists of the principal Docker images which are available at [4]:

- Metric-Platform – Image with the components of the Metric Platform and some metric providers developed in CROSSMINER. This image can be run as different containers;
- Metric-Platform Database (oss-db) – MongoDB image to store the data of the metric platform. Currently it runs as one container but may be replaced by a replication or shared MongoDB cluster if needed;

- CROSSMINER Admin – a web application developed to administrate the CROSSMINER platform e.g., to create projects, configure analysis tasks, etc;
- Knowledge Base Service – Image aggregating the components implementing the Knowledge Base. It is used to run the cross-project analyses and provides the API to give access computed data;
- Knowledge Base Database (kb-db) – MongoDB image to store the data from the analyses performed by the Knowledge Base Service.
- DevOps Dashboard [6] – Image with the Perceval tool to process CROSSMINER data and providing a backend for the DevOps Dashboard. The Perceval backend is an extension to the Perceval data collection tool that will collect data from the Knowledge Base and Metric Platform databases using the API REST, and it will convert it to JSON documents.

6 Conclusion

The CROSSMINER project is an OSS project focus on increasing dependence on open-source software (OSS) and it permits to take educated decisions. The project is a follow-up to a previous project called OSSMETER. The complexity of its legacy components raised challenges related to the integration and the communications among the heterogenous built-in components. To handle all these integration issues, we explored some technologies inspired from microservice architectures to assembly those heterogenous components into a unified platform and to address the security aspects of the whole platform.
Finally, we adopted a container-based distributed strategy based on the Docker Compose technology [4] to automate the application deployment.

References

1. Alessandra Bagnato, Konstantinos Barmpis, Nik Bessis, Luis Adrián Cabrera-Diego, Juri Di Rocco, Davide Di Ruscio, Tamás Gergely, Scott Hansen, Dimitris S. Kolovos, Philippe Krief, Ioannis Korkontzelos, Stéphane Laurière, Jose Manrique Lopez de la Fuente, Pedro Maló, Richard F. Paige, Diomidis Spinellis, Cedric Thomas, Jurgen J.Vinju: Developer-Centric Knowledge Mining from Large Open-Source Software Repositories (CROSSMINER). STAF Workshops 2017: 375-384 Marburg (2017).
2. Bruno Almeida, Sophia Ananiadou, Alessandra Bagnato, Alberto Berreteaga Barbero, Juri Di Rocco, Davide Di Ruscio, Dimitrios S. Kolovos, Ioannis Korkontzelos, Scott Hansen, Pedro Maló, Nikolaos Drivalos, Richard F. Paige, Jurgen J. Vinju: OSSMETER: Automated Measurement and Analysis of Open Source Software. STAF Projects Showcase 2015: 36-43 L'Aquila (2015).
3. CROSSMINER SCAVA GitHub Repository, last accessed 07/06/2019: https://github.com/crossminer/scava.
4. CROSSMINER SCAVA-Deployment GitHub Repository, last accessed 07/06/2019: https://github.com/crossminer/scava-deployment/tree/dev
5. CROSSMINER WIKI: https://scava-docs.readthedocs.io/en/latest/, last accessed 07/06/2019.
6. Perceval: https://crossminer.biterg.io/app/kibana#/dashboard/ScavaProject /, last accessed 07/06/2019.