Data streaming architecture based on Apache Kafka and GitHub for tracking students' activity in higher education software development courses

Milan Miloradović

Department for e-business
The University of Belgrade, Faculty of organizational sciences
Belgrade, Serbia
mm20213510@student.fon.bg.ac.rs
https://orcid.org/0000-0003-4101-2350

Ana Milovanović

Department for e-business
The University of Belgrade, Faculty of organizational sciences
Belgrade, Serbia
am20205027@student.fon.bg.ac.rs
https://orcid.org/0000-0001-5282-881

Abstract — Data streaming architecture can be used to collect data and gain insights into the dynamics of individual or collaborative software development activity that takes place in higher education courses. There is a place to further investigate streaming architecture in a given context. The code versioning platforms, such as GitHub, serving as data sources in the existing implementations of data streaming architecture are lacking in practice. The goal of this paper is to investigate the implementation of a custom data streaming architecture that could be used to track real-time students' analytics in higher education software development courses. The solution is based on Apache Kafka and GitHub platforms. Also, the architecture developed in the paper could be considered when planning on integrating LMS (Learning Management System) as a visual web interface for students' analytics.

Keywords — data streaming architecture, Apache Kafka, GitHub, higher education, software development, Learning Management System (LMS)

I. INTRODUCTION

Data streaming architecture is based on the concept of events [1]. Event Streaming Platforms, which are based on the data streaming architecture, provide the infrastructure that enables software to react in real-time to the given events [1]. Apache Kafka is an open-source streaming platform that can make use of producers which are applications that are sending messages to the Kafka broker [2]. Kafka Broker stores messages that are later accessed by consumers [2].

Git represents a system that enables tracking changes to the user files and it is considered a Version Control System (VCS) [3]. GitHub platform relies on git and its' commands to perform version control of user files.

The aim of this paper is to provide a possible solution to the data steaming architecture that would be used to collect and process data on students' activity that takes place on version control platforms in higher education software development courses.

Upon considering different use cases of implementing data streaming architecture and taking good architectural practices into account, possible architecture is formed. The purpose of this paper is to investigate the integration of the version control platform (GitHub platform) with the data streaming platforms such as Kafka to process events generated while conducting higher education software engineering courses. This use case is lacking in practice and could be beneficial for further related work as far as higher education is concerned.

The solution gives a presentation of a data streaming architecture based on Apache Kafka and the GitHub platform. Besides, GitHub webhook concept is described, as well as the flow of communication between Kafka producer and Kafka consumer. In the proposed solution, the communication between Kafka producer and Kafka consumer starts with a GitHub webhook event.

Kafka producer and Kafka consumer are implemented using the Java Spring Boot framework. Java Spring Boot is an open-source, microservice-based Java web framework [4]. The microservice architecture provides developers with a fully enclosed application, including embedded application servers [4].

The question of integrating Learning Management systems (LMS) such as Moodle LMS into the architecture for a unified dashboard preview of students' analytics is also considered.

II. LITERATURE REVIEW

As the authors state in [5], the GitHub platform provides insights into social coding activities. Apart from the popular usage of the GitHub platform in the software development industry, this is also the reason to consider using GitHub as a collaborative software development platform [6] in higher education setup and as a data source in data streaming architecture.

GitHub data analysis done in [7] demonstrates the possibilities of data generated through events on the GitHub platform. Different analytics are considered including the number of commits per contributor and SNA (Social Network Analysis) analysis [7]. Those analytics could also be considered when implementing Kafka consumer.

Some of the research papers that are dealing with the integration of code versioning platforms into the curriculum of the software development university courses have relied on the GitHub platform and its' functionality to gain insight into students' activities [8][9][10]. However, using GitHub as a data source provider to deal with real-time stream processing and analytics in an educational environment is lacking in practice.

New streaming technologies that are available today handle stream data with high performance with a message throughput of millions of messages per second [2]. Data platforms handle data from different sources and stream data to different consumers [2].

Events in the Kafka ecosystem are assigned to topics [11]. Those topics are holding different numbers of logs (shards or partitions). The number of shards is configurable, thus scalability in the Kafka ecosystem is provided [11].

The use cases of deploying data streaming architecture are quite diverse in education. In [12] authors have developed a cloud-based e-learning platform to provide educational content for the agricultural community. The streaming analysis here is employed in real-time using Apache Kafka and Apache Spark to provide high-quality video content to users and to control server resources.

A popular area to employ streaming platforms is certainly IoT (Internet of Things). In the research presented in [13], the authors developed a course that is called Network-of-Things Engineering Lab (NoteLab). The course combines IoT, edge and cloud computing and is dealing with the implementation of interfaces and protocols that connect the entire system (MQTT, COAP and HTTP) [13]. Kafka is used as a connector, consisting of a Kafka broker among other components, using publish/subscribe protocol to connect with Kafka producers and Kafka-Firebase connector to connect to Firebase real-time database [13].

Machine learning is another popular area to take into consideration when investigating the usage of streaming platforms. In [14] Apache Kafka is utilized to implement a stream processing system based on a publish-subscribe pattern. The system developed in [14] deals with vehicle detection based on its' attributes such as color, speed and type. There are two main steps, the first being getting the vehicle information from a video, and the second step is streaming that information to subscribers.

In [15] by making use of DevOps and cloud computing "continuous quality assurance approach" is facilitated [15]. The prototype of the solution consisted of detecting problems that are likely to happen when the new versions of the microservices are being in building state, deployed, or targeted with requests [15]. The data crawler observes all microservice containers that are running, collects the relevant data and sends the data to the Apache Kafka cluster. Data is further being fetched and indexed in Elasticsearch [15].

Another usage of data streaming is implemented in CERN HSE (occupational Health & Safety and Environmental protection) Unit [16] which deals with the implementation of the CERN Safety Policy. Researchers developed REMUS (Radiation and Environmental Unified Supervision) system that is using an open-source Apache Kafka streaming platform to stream real-time data to their Web Interfaces and Data Visualization Tools [16].

III. METHODOLOGY

In order to create custom data streaming architecture based on Apache Kafka and GitHub for tracking students' activity, GitHub webhook, Kafka producer and consumer are used.

It is necessary to identify components of the data streaming architecture. Identified components are as follows (Table 1):

- GitHub users
- GitHub webhook
- Kafka producer
- Kafka cluster
- Kafka consumer

Kafka producer and consumer are implemented using the Java Spring Boot framework.

Table 1. Identified Components

No.	**Data streaming architecture**	
	Components	*Example*
1.	GitHub user	Student working on GitHub repository (push event).
2.	GitHub webhook	Mechanism integrated into GitHub organization of higher education institution (elab).
3.	Kafka producer	The application that broadcasts messages.
4.	Kafka cluster	Its' task is to process and organize the data.
5.	Kafka consumer	The application that receives messages.

The first part of the architecture is the GitHub platform. As already stated, it is a platform for version control and is usually used for various social coding [5] activities. In this particular case, the GitHub platform serves to broadcast events that are created within the GitHub organization. Some of the event types that should be considered are: push, pull request, merge, view, commit, etc.

GitHub users, in this particular example, are students within the GitHub organization, which generate events over their repositories. When one of these events happens, the GitHub webhook [17] is triggered as shown in Fig. 1.

After that, the GitHub webhook forwards the POST request to the Kafka producer, which further determines to which topic to send the event.

The second part of the architecture is the Apache Kafka (Fig. 1). Apache Kafka consists of a Kafka cluster, Zookeeper server, as well as producer and consumer. Kafka producer and Kafka consumer are applications where one application broadcasts (producer) messages and the other one receives messages from the broadcaster (consumer) [18].

Kafka cluster's task is to process and organize the data that is passed to it. Each Kafka cluster contains a list of topics, where incoming messages are redirected. A topic is made up of one or more partitions, and each partition is an edited, immutable sequence to which new records are constantly added [18].

Based on the GitHub event type that is being processed by the Kafka producer, messages could be passed to a different topic. For example, for each commit event being processed, the Kafka producer sends a message to the topic Topic 1 (Fig. 1). For each pull event, the Kafka producer can be configured to send a message to the topic Topic 2 (Fig. 1) and for each merge event to the topic Topic 3 (Fig. 1).

Identified topics are also given in the table (Table 2):

Table 2. Identified Topics

No.	**Kafka topics**	
	Topics	*Example*
1.	Topic 1	Topic which contains only commit events.
2.	Topic 2	Topic which contains only pull events.
3.	Topic 3	Topic which contains only merge events.

Kafka broker is an instance of a Kafka cluster. It receives messages from the producer, assigns an offset to each message, and then stores those messages on a disk [18].

Zookeeper is a site management and performance monitoring tool in the Kafka cluster [18].

In the end, the Kafka consumer application reads data from the topic and students' activity analytics are calculated. Later these analytics could be presented in some web interface such as dashboard preview.

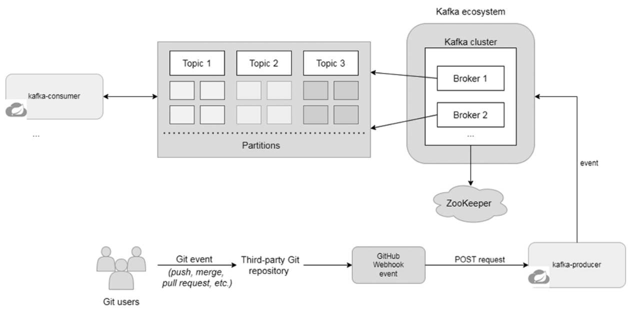
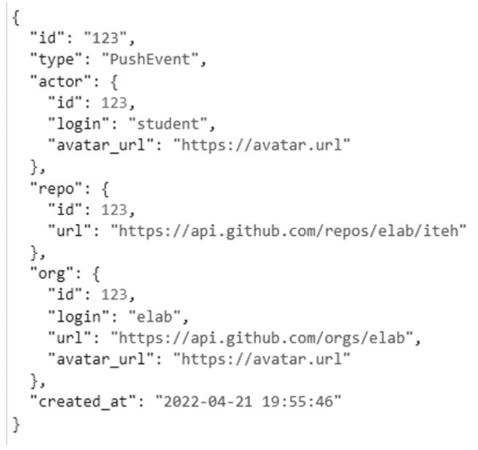


Fig. 1. Proposed Data streaming architecture diagram

An example (TABLE I) of an event, which is triggered from a GitHub webhook, is shown in the figure (Fig. 2).

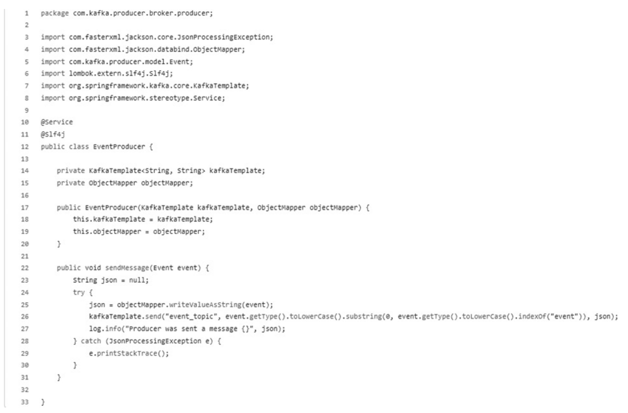


```
{
  "id": "123",
  "type": "PushEvent",
  "actor": {
    "id": 123,
    "login": "student",
    "avatar_url": "https://avatar.url"
  },
  "repo": {
    "id": 123,
    "url": "https://api.github.com/repos/elab/iteh"
  },
  "org": {
    "id": 123,
    "login": "elab",
    "url": "https://api.github.com/orgs/elab",
    "avatar_url": "https://avatar.url"
  },
  "created_at": "2022-04-21 19:55:46"
}
```

Fig. 2. Push event

Push event (Fig. 2) contains data about event id, type, actor (in this case the student who generated the event), repository (repo), GitHub organization (org), and date (created_at) when the event is created.

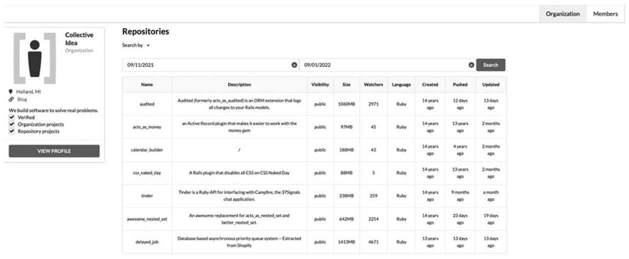
Kafka's producer's task is to forward push events to a certain topic (event_topic) as shown in Fig. 3. Once the data arrives at the topic, the Kafka consumer reads the data. It is possible that the consumer application, upon reading the data, performs analytics and presents the data through the web interface. The web interface could be integrated with Moodle LMS, but this option needs to be investigated further.



```java
package com.kafka.producer.broker.producer;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.kafka.producer.model.Event;
import lombok.extern.slf4j.Slf4j;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;

@Service
@Slf4j
public class EventProducer {

    private KafkaTemplate<String, String> kafkaTemplate;
    private ObjectMapper objectMapper;

    public EventProducer(KafkaTemplate kafkaTemplate, ObjectMapper objectMapper) {
        this.kafkaTemplate = kafkaTemplate;
        this.objectMapper = objectMapper;
    }

    public void sendMessage(Event event) {
        String json = null;
        try {
            json = objectMapper.writeValueAsString(event);
            kafkaTemplate.send("event_topic", event.getType().toLowerCase().substring(0, event.getType().toLowerCase().indexOf("event")), json);
            log.info("Producer was sent a message {}", json);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

}
```

Fig. 3. Kafka producer implementation

Fig. 4 and Fig. 5 show the web interface. Fig. 4 shows the profile of GitHub organization, as well as a list of its repositories. Besides that, the figure also shows repository search by date.

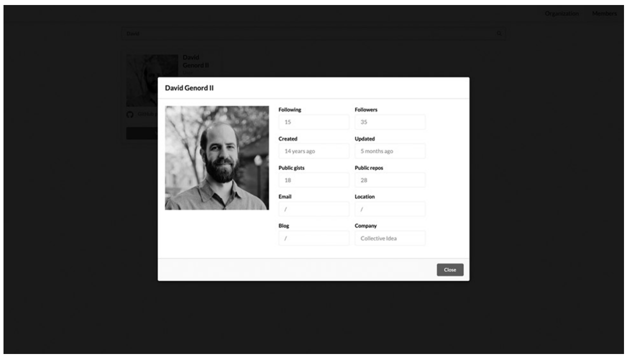


09/11/2021 09/01/2022

Name	Description	Visibility	Size	Watchers	Language	Created	Pushed	Updated
audited	Audited (formerly acts_as_audited) is an ORM extension that logs all changes to your Rails models.	public	1060MB	2971	Ruby	14 years ago	12 days ago	13 days ago
acts_as_money	an Active Record plugin that makes it easier to work with the money gem	public	97MB	45	Ruby	14 years ago	13 years ago	2 months ago
calendar_builder	/	public	188MB	43	Ruby	14 years ago	4 years ago	2 months ago
css_naked_day	A Rails plugin that disables all CSS on CSS Naked Day	public	88MB	5	Ruby	14 years ago	13 years ago	2 months ago
tinder	Tinder is a Ruby API for interfacing with Campfire, the 37Signals chat application.	public	238MB	259	Ruby	14 years ago	9 months ago	a month ago
awesome_nested_set	An awesome replacement for acts_as_nested_set and better_nested_set.	public	642MB	2254	Ruby	14 years ago	23 days ago	19 days ago
delayed_job	Database based asynchronous priority queue system -- Extracted from Shopify	public	1413MB	4671	Ruby	13 years ago	13 days ago	13 days ago

Fig. 4. Repository search by date

Fig. 5 shows the review of a member's profile within GitHub organization. Some of the data are: number of followers, number of public repositories, location, email, company, etc.



David Genord II

Following: 15	Followers: 35
Created: 14 years ago	Updated: 5 months ago
Public gists: 18	Public repos: 28
Email: /	Location: /
Blog: /	Company: Collective Idea

Fig. 5. Review of a member's profile within GitHub organization

IV. CONCLUSION AND IMPLICATIONS

In conclusion, it is important to state that there is a possible way to implement data streaming architecture which is based on Apache Kafka and GitHub platform for tracking students' activity in higher education software development courses.

GitHub webhook mechanism is used to integrate the GitHub platform and Kafka producer application. Students are working on their individual or collaborative GitHub repositories and thus are generating events. GitHub webhook is triggered by some event types that are defined when creating a webhook. Upon certain GitHub events happening, the webhook is sending the POST requests to the Kafka producer. The producer further determines to which topic to send the event. In the end, the Kafka consumer reads the data from the topic and calculates analytics based on students' activity.

The given solution could be used as a starting point for further consideration of this idea. The main goal of implementing an architecture that relies on Apache Kafka is to obtain the necessary analytics on students' activity generated on the GitHub platform. The advantage of this solution is in the speed of message processing and in the number of messages that are processed. This provides teachers with the latest information on students' activities in real-time.

What could be further investigated is the question of integrating LMS (Learning Management System) such as Moodle with the Kafka consumer application. This would allow real-time web presentation of data that is analytically processed before. The dashboard presented in the LMS solution could provide different analytics and previews customized to LMS user that has access to it. Different dashboard data could be presented to the students and to the teachers depending on the role that is given to them.

REFERENCES

[1] Confluent Documentation, "Welcome to Event Streaming Patterns," https://developer.confluent.io/patterns, Apr. 19, 2022.

[2] T. Dunning and E. Friedman, Streaming architecture: new designs using Apache Kafka and MapR streams. O'Reilly Media, Inc., 2016.

[3] S. Chacon and B. Straub, Pro Git. Berkeley, CA: Apress, 2020. doi: 10.1007/978-1-4842-0076-6.

[4] Spring Docs, "'Why Spring?,'" https://spring.io/why-spring, Apr. 19, 2022.

[5] M. AlMarzouq, A. AlZaidan, and J. AlDallal, "Mining GitHub for research and education: challenges and opportunities," International Journal of Web Information Systems, vol. 16, no. 4. Emerald Group Holdings Ltd., pp. 451–473, Jun. 29, 2020. doi: 10.1108/IJWIS-03-2020-0016.

[6] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, and W. Wang, "The Emergence of GitHub as a Collaborative Platform for Education," Feb. 2015. doi: 10.1145/2675133.2675284.

[7] A. Milovanović, D. Stojanović, and D. Barać, "Exploring Possibilities of Integrating Version Control Platforms in Higher Education Through GitHub Data Analysis," J. Women's Entrep. Educ., vol. 2021, no. 3–4, pp. 113–133, Dec. 2021, doi: 10.28934/jwee21.34.pp113-133.

[8] M. D. Beckman, M. Çetinkaya-Rundel, N. J. Horton, C. W. Rundel, A. J. Sullivan, and M. Tackett, "Implementing Version Control With Git and GitHub as a Learning Objective in Statistics and Data Science Courses," J. Stat. Data Sci. Educ., vol. 29, no. sup1, Mar. 2021, doi: 10.1080/10691898.2020.1848485.

[9] A. Arroyo, M. Ramos Montes, and J. D. Segrelles Quilis, "A Pilot Experience with Software Programming Environments as a Service for Teaching Activities," Appl. Sci., vol. 11, no. 1, Dec. 2020, doi: 10.3390/app11010341.

[10] J. Feliciano, M.-A. Storey, and A. Zagalsky, "Student experiences using GitHub in software engineering courses," May 2016. doi: 10.1145/2889160.2889195.

[11] M. Kleppmann, "Thinking in events: from databases to distributed collaboration software," in Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems, 2021, pp. 15–24.

[12] J.-H. Chang, P.-S. Chiu, and C.-F. Lai, "Implementation and evaluation of cloud-based e-learning in agricultural course," Interact. Learn. Environ., pp. 1–16, Sep. 2020, doi: 10.1080/10494820.2020.1815217.

[13] J. Dizdarević and A. Jukan, "Engineering an IoT-Edge-Cloud Computing System Architecture: Lessons Learnt from An Undergraduate Lab Course," in 2021 International Conference on Computer Communications and Networks

(ICCCN), 2021, pp. 1–11.
[14] S. Kul, I. Tashiev, A. Şentaş, and A. Sayar, "Event-based microservices with Apache Kafka streams: A real-time vehicle detection system based on type, color, and speed attributes," IEEE Access, vol. 9, pp. 83137–83148, 2021.
[15] F. Oliveira et al., "Delivering software with agility and quality in a cloud environment," IBM J. Res. Dev., vol. 60, no. 2–3, pp. 10–11, 2016.
[16] A. Ledeul, G. Segura Millan, A. Savulescu, B. Styczen, and G. Switzerland, "Data streaming with Apache Kafka for CERN Supervision, Control and Data Acquisition System for Radiation and Environmental Protection," 2019, doi: 10.18429/JACoW-ICALEPCS2019-MOMPL010.
[17] GitHub docs, "Webhooks and events," https://docs.github.com/en/developers/webhooks-and-events/about-webhooks, May 14, 2021.
[18] Confluent Documentation, "Introduction to Kafka," https://developer.confluent.io/what-is-apache-kafka/, Apr. 19, 2022.