FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Software Architecture

Jakub Pavlíček

PILSEN, CZECH REPUBLIC

5. 4. 2025

Contents

Introduction 1

1.1 Purpose

This document presents a comprehensive overview of the software architecture for a pump control and Application Lifecycle Management (ALM) data management system. The architecture is designed to provide a robust, scalable, and user-friendly solution for managing pump operations, such as extracting structured project data, and integrating it into the SPADe system for further anti-pattern analysis.

1.2 System Overview

The system is a complex software solution that is responsible for:

- Extracting data from various ALM tools (Jira, Git, Github, etc.).
- Mapping and storing extracted data into a provided MySQL database.
- Providing a web-based user interface to configure pump behavior.
- Enabling real-time notifications of the pumping process with a message broker.
- Ensuring easy deployment across different environments using containerization.

1.3 Primary Objectives

The primary objectives of the systems include:

- Reliable pump operation, supporting multiple ALM tools with seamless integration.
- Compatibility with existing ALM platforms and full integration with the SPADe data model.
- Scalability and modularity, allowing easy extension to support additional ALM tools.
- Pseudonimized data storage, ensuring compliance with privacy and security requirements, such as NDA.
- User-friendly interface for easy pump configuration and real-time monitoring.

- Real-time data streaming for monitoring and processing extraction progress.
- Containerized deployment to ensure consistent deployment across various environments.

Architecture overview 2

2.1 High-Level Architecture Overview

The frontend is built with React, providing a user-friendly interface for interacting with the entire system. The user starts the data extraction by sending a request to the backend API, which responds with a HTTP status `202 Accepted`, indicating that the request was successfully received. The core component of the system is the backend Spring Boot API, which initiates the data extraction process from various ALM tools using the proper type of data pump. Once the data has been extracted, it is mapped to the MySQL database, either according to the default configuration or user-specified mapping. Since the whole pumping process is an asynchronous process, the backend also manages message broker Apache Kafka, that notifies the frontend application about the status of the extraction. The architecture of the proposed system can be seen in the figure 2.1.

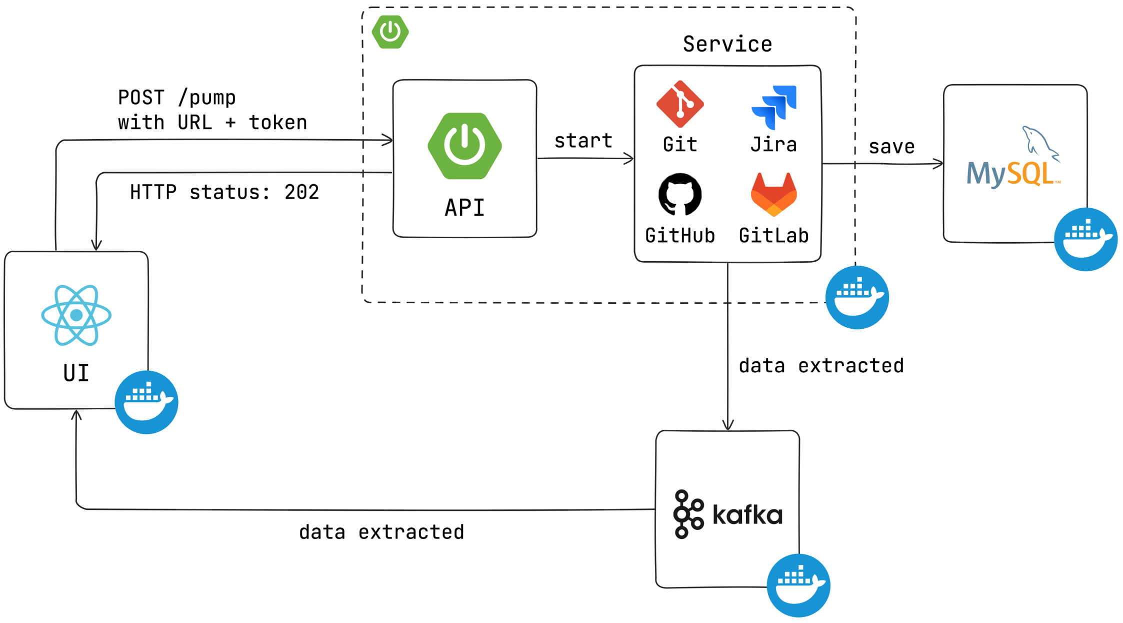

Figure 2.1: High-Level Architecture Diagram

2.2 Technology Stack

- **Frontend** – React.js, 19.0.0
- **Backend** – Java, 23.0.2 + Maven, 3.9.9 + Spring Boot, 3.4.4
- **Database** – MySQL, 9.2.0
- **Event Streaming** – Apache Kafka, 4.0.0
- **Containerization** – Docker, 27.2.0
- **Env file sharing** – git-crypt, 0.7.0

Component Detailed Description 3

3.1 User Interface (UI Layer)

Technology: React.js

Responsibilities:

- Provide an intuitive pump control interface to manage data extraction.
- Display real-time system status updates.
- Handle user interactions and inputs.
- Communicate with backend services.

Communication Channels:

- REST API for handling requests.
- Apache Kafka for streaming real-time updates.

3.2 API Layer

Technology: Spring Boot

Responsibilities:

- Provide a standardized interface for communication between the frontend and backend services.
- Validate and process the requests from the frontend application.
- Route requests to the appropriate pump services for data extraction.

Key Features:

- Request and response handling.
- Error handling.

3.3 Service Layer

Technology: Spring Boot

Responsibilities:

- Implement core business logic for pump control operations.
- Manage pump control operations, such as data extraction and transformation.
- Enforce safety protocols and secure operations.
- Publish events for system updates.

Key Processes:

- Command processing and execution.
- Data anonymization.
- Event publishing via Apache Kafka.
- Error management.

3.4 Database Layer

Technology: MySQL

Purpose: Persistent storage of ALM data

Stored Information:

- Software project details.
- Development lifecycle data.
- People involved in the project.

3.5 Event Streaming

Technology: Apache Kafka

Responsibilities:

- Asynchronous communication between the frontend and backend.
- Real-time status updates for frontend.
- Decoupling of system components.

Use Cases:

- Pump status broadcasting.
- Error notifications.

3.6 Containerization

Technology: Docker & Docker Compose

Benefits:

- Ensures consistent environments across development, test, and production.
- Simplifies deployment and dependency management.
- Isolates components for better troubleshooting.
- Enables easy scaling of services.

Configuration Management:

- Handles service dependencies.
- Allocates resources efficiently.
- Configures network.

List of Figures