Collection description

Tomasz Olejarczuk

Fontys

20-06-2024

Contents

Collection description

Teamium project

Introduction

My involvement in the Teamium project began mid-way through its development cycle. While I didn't participate in the initial phases, I quickly integrated into the team and focused my efforts on addressing key technical challenges related to the deployment, optimization, and communication of the core MATLAB algorithm.

My primary contributions to the Teamium project revolved around:

- **MATLAB Algorithm Deployment Optimization:** I thoroughly analyzed and implemented different deployment strategies for the MATLAB algorithm, including the use of MATLAB Production Server and Standalone Archives. This exploration aimed to address licensing constraints, improve performance, and enhance the algorithm's integration with the rest of the system.
- **CI/CD Pipeline Development:** I designed and implemented a robust Continuous Integration/-Continuous Delivery (CI/CD) pipeline using GitHub Actions. This automated the building, versioning, and deployment of the MATLAB algorithm, streamlining the development process and ensuring that the latest version was always readily available.
- **API Wrapper Development and Integration:** I played a key role in developing a C# ASP.NET Core API wrapper to act as an interface between external applications and the MATLAB server. This involved exposing endpoints, managing data storage (using MinIO and PostgreSQL), and facilitating real-time communication through WebSockets.
- **Integration of RabbitMQ and Redis:** I explored and implemented the integration of RabbitMQ as a message broker and Redis as a cache, enhancing the system's asynchronous communication capabilities, improving performance, and ensuring data reliability.

Throughout my work on the Teamium project, I embraced the opportunity to learn and apply new technologies, such as Docker, RabbitMQ, MinIO, and Github Actions. I also focused on producing comprehensive documentation to ensure the project's maintainability and facilitate knowledge transfer within the team. My contributions helped to address critical challenges in the project's architecture, enhance its scalability, and improve its overall reliability and performance.

The following sections of this documentation will detail my specific contributions and learning experiences during each sprint of the project.

Sprints

Sprint 1: Joining the formed group

Overview

During my initial sprint, the primary objective was to quickly get up to speed with the existing project's goals, architecture, and challenges. This involved understanding the MATLAB algorithm's role, the current deployment methods, and identifying areas for improvement.

Key Tasks

- **Onboarding and Knowledge Transfer:** Actively engaged with team members and stakeholders to gain a comprehensive understanding of the project's requirements, current progress, and technical aspects.
- **Code Review and Analysis:** Performed a thorough review of the existing MATLAB codebase to identify potential optimizations, performance bottlenecks, and areas where deployment strategies could be enhanced.
- **Research and Exploration:** Explored different MATLAB deployment options, including MATLAB Production Server and Standalone Archives, to determine the most suitable approach for the project's needs.

Evidence

- [No directly linked evidence as this sprint was focused on onboarding and project familiarization]

Personal Reflections

Joining a project in progress presented a unique challenge, as I needed to quickly adapt to the existing team dynamics and codebase. This sprint was crucial for establishing a strong foundation for my subsequent contributions, as it allowed me to identify key areas where my expertise could make a significant impact. The in-depth code review and research into deployment strategies were instrumental in shaping my approach for the following sprints.

Learning Points

- **Effective Onboarding:** The importance of active communication and collaboration with team members and stakeholders during the initial onboarding phase to gain a deep understanding of

the project context.
- **Technical Due Diligence:** The value of thoroughly reviewing existing code and documentation before proposing or implementing changes.
- **Openness to Exploration:** The benefits of researching and considering various technical approaches before committing to a specific solution.

Sprint 2: MATLAB Deployment Optimization and CI/CD Pipeline Development

Overview

Building upon the initial analysis, Sprint 2 focused on optimizing the MATLAB algorithm's deployment and establishing a streamlined CI/CD pipeline. The main goals were to address licensing constraints, improve scalability, and automate the build and deployment process.

Key Tasks

- **MATLAB Production Server Analysis and Implementation:** Conducted a comprehensive evaluation of MATLAB Production Server as a deployment solution, comparing its benefits and drawbacks to alternatives like MATLAB Standalone Archives. Produced detailed documentation outlining the analysis and recommendations.
- **MATLAB Standalone Archives Analysis and Implementation:** Explored the use of MATLAB Standalone Archives within a Dockerized environment to address licensing limitations and streamline deployment. Created a MATLAB build script to automate the creation of the archive and packaged it into a Docker image. Produced a detailed analysis document outlining the advantages and disadvantages of this approach.
- **CI/CD Pipeline Implementation for MATLAB Algorithm Deployment:** Designed and implemented a robust CI/CD pipeline using GitHub Actions to automate the building, versioning, and deployment of the MATLAB algorithm. Incorporated semantic versioning and produced comprehensive documentation detailing the pipeline's steps and potential future deployment strategies.

Evidence

- [@Analysis and Considerations for Using MATLAB Production Server]
- [@Analysis and Considerations for Using MATLAB Standalone Archives]
- [@CI/CD Pipeline for MATLAB Algorithm Deployment]

Personal Reflections

Sprint 2 was a period of intense learning and exploration, as I delved into the intricacies of MATLAB deployment strategies and CI/CD pipeline development. This involved a steep learning curve in understanding MATLAB's unique features and capabilities. Through extensive research, analysis, and hands-on experimentation, I was able to successfully address the project's licensing challenges, enhance deployment efficiency, and establish a foundation for future scalability.

Learning Points:

- **Trade-offs in Technology Selection:** The importance of weighing the pros and cons of different technological solutions (MATLAB Production Server vs. Standalone Archives) to find the best fit for the project's specific needs.
- **Automation's Impact:** The significant benefits of CI/CD pipelines in streamlining development workflows, reducing manual errors, and enabling rapid, reliable deployments.
- **Scalability Considerations:** The need to consider future growth and scalability requirements when designing deployment solutions.
- **The Reality of Legacy Technologies:** Sometimes, projects require working with technologies that might not be the most modern or enjoyable. It's essential to assess the trade-offs and focus on delivering results while minimizing long-term commitment to such technologies.
- **The Importance of Documentation:** Thorough documentation is crucial for understanding complex systems, making informed decisions, and ensuring knowledge transfer within a team, especially when dealing with technologies that may not be widely familiar.

Sprint 3: Adding API Wrapper and integrating MinIO, PostgreSQL and RabbitMQ

Overview

Sprint 3 was a crucial phase of the project, focusing on integrating essential components for data storage, log management, inter-service communication, and establishing a robust foundation for the entire system. This involved not only integrating external technologies (MinIO, PostgreSQL, and RabbitMQ) but also developing a crucial C# API wrapper to act as the interface between the MATLAB server and external applications. Additionally, this sprint included the exploration of DAPR as a potential future enhancement to improve the system's architecture and scalability.

Key Tasks

- **Development of C# API Wrapper:**

  - Designed and implemented a C# ASP.NET Core controller to act as an API wrapper for interacting with the MATLAB Production Server.
  - Exposed API endpoints for querying MATLAB, retrieving results (including heatmap images), and managing data stored in MinIO and PostgreSQL.
  - Implemented comprehensive error handling and logging mechanisms to ensure the API's stability and reliability.

- **MinIO Integration for Heatmap Image Storage:** Implemented MinIO, an S3-compatible object storage solution, to store and retrieve heatmap images generated by the MATLAB server. The C# API wrapper was configured to interact with MinIO using the MinIO SDK for .NET.
- **PostgreSQL Integration for Log and Communication Data Storage:** Integrated PostgreSQL as the central database for storing logs generated by the MATLAB server, incoming requests, and corresponding responses. A comprehensive database schema was designed, and Entity Framework Core was used to facilitate seamless data interactions within the C# API wrapper.
- **RabbitMQ Integration for Asynchronous Communication and Message Queuing:** Explored and planned the integration of RabbitMQ as a message broker to facilitate asynchronous communication between the API wrapper and MATLAB server. This involved designing the message flow, selecting appropriate queues, and configuring the Docker environment for RabbitMQ deployment.
- **WebSocket Communication for Real-Time Progress and Logging:** Implemented WebSocket communication between the C# API wrapper and MATLAB server to provide real-time progress updates and log streaming to clients. This involved establishing WebSocket connections, handling progress messages, and transmitting logs in real time.
- **Enhancing MATLAB Production Server with DAPR:** Explored the potential benefits of integrating the Distributed Application Runtime (DAPR) into the MATLAB Production Server API wrapper project. Outlined a roadmap for DAPR adoption, considering the existing architecture and requirements.

Evidence

- [@API Wrapper for MATLAB Production Server Integration]
- [@MinIO Integration for Heatmap Image Storage]
- [@PostgreSQL Integration for Log and Communication Data Storage]
- [@RabbitMQ Integration for Asynchronous Communication and Message Queuing]
- [@WebSocket Communication for Real-Time Progress and Logging]
- [@Enhancing MATLAB Production Server with DAPR]

Personal Reflections

Sprint 3 was an intense period focused on building the core infrastructure of the project. Developing the API wrapper was a central task, as it was the backbone for integrating all the other components and enabling external applications to interact with the MATLAB server. The experience highlighted the importance of creating a well-designed and robust API that could handle complex data exchanges, manage various storage mechanisms, and provide real-time feedback to users. The integration of multiple technologies also underscored the need for careful planning, meticulous configuration and thorough testing to ensure seamless operation. Furthermore, the exploration of DAPR opened up exciting possibilities for enhancing the system's architecture and scalability in the future.

Learning Points

- **API Design and Development:** The importance of designing a well-structured, maintainable, and scalable API that can effectively handle complex data exchanges and interactions with various backend systems.
- **Data Storage Strategies:** The importance of selecting appropriate storage solutions for different types of data (e.g., object storage for images, relational databases for structured data) based on their specific characteristics and access patterns.
- **Asynchronous Communication:** The benefits of using message brokers like RabbitMQ to decouple components, improve scalability, and enhance fault tolerance in distributed systems.
- **Integration Challenges:** The complexities involved in integrating different technologies, including configuration, dependency management, and ensuring seamless communication between components.
- **Real-time Communication:** The power of WebSockets in providing real-time updates and enhancing user experiences in web applications.
- **Exploring New Technologies:** The value of proactively exploring new technologies like DAPR, even before their full implementation, to understand their potential benefits and how they might fit into the project's future roadmap.

Sprint 4: Finalizing Integrations and Optimizing Deployment

Overview

Sprint 4 marked a crucial milestone in the Teamium project, as we focused on solidifying the integration of various components and optimizing the deployment process. Key achievements during this sprint included:

- **Finalizing the C# API Wrapper:** Completed the API wrapper, thoroughly tested its endpoints using Insomnia, and documented its functionality for future reference and ease of use.
- **Integrating Redis as a Cache:** Successfully implemented Redis as a caching layer between RabbitMQ and PostgreSQL, enhancing the system's performance and scalability for log storage.
- **Refining MATLAB Logging:** Overhauled the MATLAB logging mechanism to leverage RabbitMQ, ensuring reliable and asynchronous log transmission while decoupling it from the core algorithm logic.
- **Optimizing Deployment with os-shell:** Introduced the `bitnami/os-shell` container into the Docker Compose setup to automate the management of file permissions within shared volumes, streamlining the deployment process and preventing potential errors.

Key Tasks

- **Redis Cache Implementation:** Integrated Redis as a cache between RabbitMQ and PostgreSQL to buffer log messages, reduce database load, and improve overall system responsiveness.
- **MATLAB Logging Rewrite:** Rewrote the MATLAB logging mechanism to leverage RabbitMQ, ensuring reliable message delivery and improving the system's architecture.
- **Docker Compose Refinement:** Incorporated the `bitnami/os-shell` container into the Docker Compose file to automate the setup of file permissions in shared volumes.

Evidence

- @Docker Compose Configuration for Teamium Project
- @Insomnia : A Critical Tool for Teamium API Development and Testing
- @Integrating Redis as a Cache for RabbitMQ Log Streaming to PostgreSQL
- @MATLAB Logging Overhaul and RabbitMQ Integration
- @Optimizing Container Deployment with `os-shell`
- @Wrapper Architecture

Personal Reflections

Sprint 4 brought a sense of accomplishment as we connected the various components of the Teamium project into a cohesive system. The implementation of Redis caching and the improved MATLAB logging mechanism addressed key performance and reliability concerns. Integrating the `bitnami/os-shell` container demonstrated the power of Docker Compose in automating essential setup tasks and simplifying the deployment process. Documenting the API wrapper's architecture and testing procedures was also a valuable exercise, solidifying our understanding of the system and providing a reference for future maintenance and enhancements.

Learning Points

- **Performance Optimization:** The impact of caching strategies on system performance, particularly in scenarios with high data throughput.
- **The Importance of Reliable Logging:** Ensuring robust log transmission and storage mechanisms is crucial for debugging, monitoring, and understanding system behavior.
- **Docker Compose Automation:** The power of Docker Compose in orchestrating complex multi-container environments and automating setup tasks.
- **Thorough Documentation:** Comprehensive documentation is essential for maintaining a complex system, onboarding new team members, and facilitating effective collaboration.