DevOps Report

Group F
(insert names)

May 2024

Contents

List of Figures

Chapter 1

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed efficitur nunc non tempor pulvinar. Quisque congue orci non porta feugiat. Morbi vitae nulla libero. Nulla a sem diam. Pellentesque ullamcorper, est sed posuere eleifend, ante elit imperdiet lorem, ac scelerisque lacus ante id ipsum. Vestibulum ac metus tempor sem mattis ornare at eget massa. Sed ut aliquam lorem.

Chapter 2

System Perspective

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed efficitur nunc non tempor pulvinar. Quisque congue orci non porta feugiat. Morbi vitae nulla libero. Nulla a sem diam. Pellentesque ullamcorper, est sed posuere eleifend, ante elit imperdiet lorem, ac scelerisque lacus ante id ipsum. Vestibulum ac metus tempor sem mattis ornare at eget massa. Sed ut aliquam lorem.

2.1 Systems dependencies

The list below, is the technolgies we are using in our application, group into categories of technogy.

Project Start - choose framework

- Microsoft AspNetCore v2: For basis web framework to handle, request, athentication,
- Dotnet API framework: For handling RESTFull API's
- Swagger: For dotnet API visualization
- Microsoft EntityFrameworkCore version 8.x: For general handling dependency injection to a database connection.
- Postgress v.8: Choosen database servere for this project.
- Docker / Docker-compose / docker-swarm: For containerized applications
- gravatar: For provide user profile images.

- Github / Github actions/ Github secrets: For storing and deploying source code
- DigitalOcean: Production server for our application

Monitoring

- prometheus/prometheus-net/prometheus-net.AspNetCore: For provide metrics to Grafana.
- Grafana: For Data Visualization of data from prometheus and direct data from Postgress database.

Logging

- Serilog AspNetCore v.8: For logging.
- Fluentd: For collecting logs from different sources.

Software quality - Testing

- Sonarqube: Static analyzer for code quality and securit issues
- CodeClimate: Another static analyzer also for code quality and securit issues
- Playwright: UI testing
- Cypress: End-2-end testing
- Xunit: For Basic Unit testing

Chapter 3

Process Perspective

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed efficitur nunc non tempor pulvinar. Quisque congue orci non porta feugiat. Morbi vitae nulla libero. Nulla a sem diam. Pellentesque ullamcorper, est sed posuere eleifend, ante elit imperdiet lorem, ac scelerisque lacus ante id ipsum. Vestibulum ac metus tempor sem mattis ornare at eget massa. Sed ut aliquam lorem.

3.1 Logging

3.1.1 fluentd

Decision Log

First we tried to implement our logging with kibana and elastic search, which worked on our local machines. The problem with this was that we only had 1GB RAM on our web server, which wasn't enough to run elastic search. To resolve that issue we looked for other technologies like fluentd.

When facing implementing logging to a system several challenges come up, for example there are disparate log formats of different applications, logs that are scattered across multiple sources (data fragmentation), reliability concerns like network outages and routing different logs to different destinations based on the application or service.

In order to manage these difficulties we decided to use fluentd. Fluentd is an open source data collector which processes the data from various sources, making it easier to manage and analyze. The following aspects show why fluentd is so powerful:

- **Unified Logging Layer:** Fluentd provides a unified platform for collecting, transforming, and routing logs from diverse sources, simplifying the log management process.
- **Data Agnosticism:** Regardless of the data source—be it Prometheus metrics, Grafana dashboards, or application logs—Fluentd seamlessly collects and processes data in a unified format, fostering interoperability and ease of analysis.
- **Configurable Routing:** With Fluentd, we could effortlessly configure log routing based on application or service, directing logs to specific destinations for centralized storage and analysis.
- **Flexibility:** Fluentd's versatility enables us to send data from any source to any destination, empowering us to adapt to evolving logging requirements seamlessly.

Implementation of fluentd

A diagram describing our use and implementation of fluentd can be seen in figure ??.

Fluentd Configuration

Fluentd is configured to listen for incoming log data via the forward input plugin on port 24224. Log data from different sources (minitwit-database, minitwit-service, prometheus, grafana) is matched using the match directive which specifies the destination path and format for the corresponding log data. Log data is buffered using the buffer directive to handle data spikes or network outages, ensuring reliable log collection.

Docker Compose Configuration

A fluentd container is defined using the fluent/fluentd:v1.12-debian image. Each service in the Docker Compose file depends on the fluentd service for logging. Logging is configured to use fluentd as the logging driver, with specific options (fluentd-async-connect, fluentd-retry-wait, fluentd-max-retries) to ensure reliable log transmission. Each service specifies a unique tag to route log data to the appropriate destination, which in our case are simple log files, in fluentd based on the matching rules defined in the fluentd.conf file.

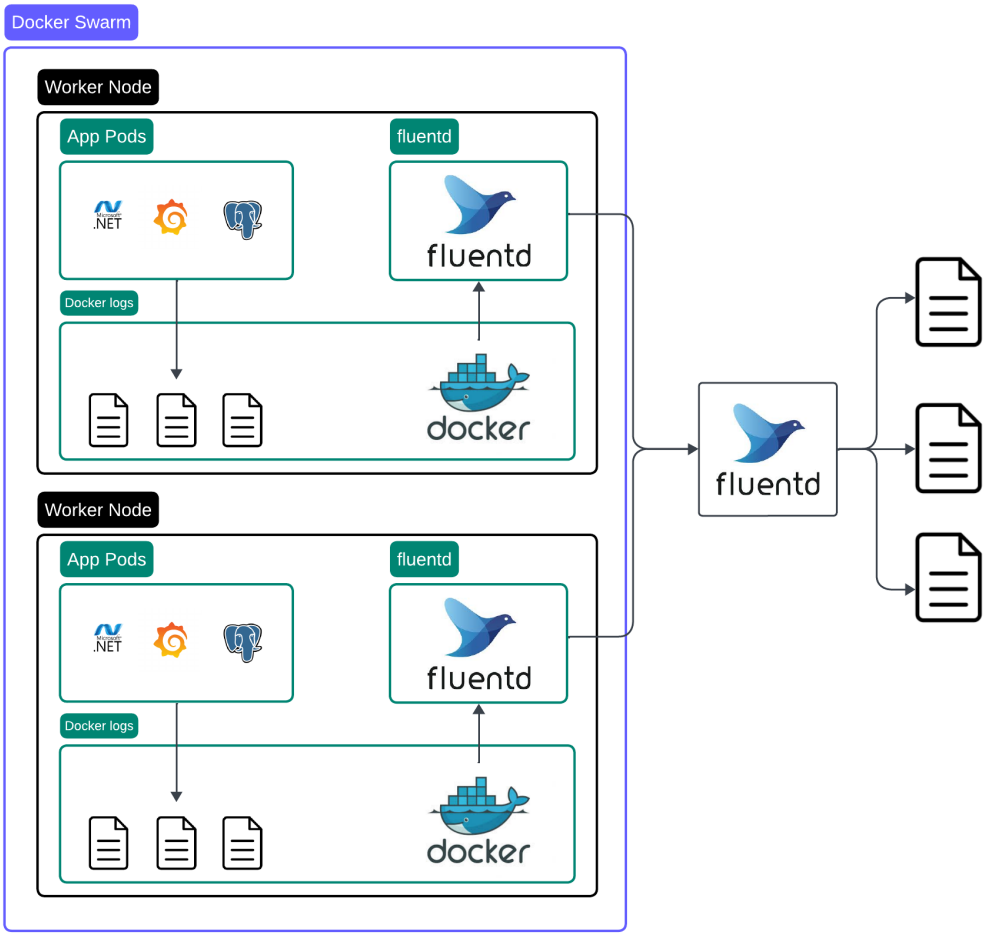

Figure 3.1: Diagram describing our implementation of fluentd

3.2 Monitoring

3.2.1 Prometheus

Decision Log

3.2.2 Grafana

Decision Log

3.3 AI-Systems used during the project

3.3.1 Reflection on ChatGPT

People would assume that using AI like ChatGPT would make developing software much easier and faster, which we think is only partially true. A positive aspect of using ChatGPT is for example it assists very well when learning a new programming language. It is much faster to learn the right syntax by asking ChatGPT rather than googling. It is also very good at explaining code, which makes it easier to understand what other people did. Another positive use is for fixing simple errors by just copying the code with the error message. Usually it resolves the mistake pretty good which also safes some time then.

The problem is when there are more complex errors, especially when it is between two or more systems. ChatGPT makes a first guess and then tends to try to find solutions within the area of the first suggestion. This on the other hand can lead to lose a lot of time because the fix of the problem lies somewhere completely else. An example for this was the implementation of our e2e tests with Cypress. When mounting the docker for cypress we had an error that it could not find the test files, which were in the correct directory. ChatGPT suggested that the volume mounting was wrong and kept suggesting to fix this issue within our compose file. However after some quick research on stackoverflow it was clear that the naming of the test files were wrong, which then could be fixed pretty quick.

As a conclusion, the use of ai-systems can safe time and make development efficient, if used in the right places. When ChatGPT doesn't give the correct answer within the first tries it is a good advice to stop using it and try to resolve the issue by your self.

Chapter 4

Lessons Learned

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed efficitur nunc non tempor pulvinar. Quisque congue orci non porta feugiat. Morbi vitae nulla libero. Nulla a sem diam. Pellentesque ullamcorper, est sed posuere eleifend, ante elit imperdiet lorem, ac scelerisque lacus ante id ipsum. Vestibulum ac metus tempor sem mattis ornare at eget massa. Sed ut aliquam lorem.

Chapter 5

Reflection and conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed efficitur nunc non tempor pulvinar. Quisque congue orci non porta feugiat. Morbi vitae nulla libero. Nulla a sem diam. Pellentesque ullamcorper, est sed posuere eleifend, ante elit imperdiet lorem, ac scelerisque lacus ante id ipsum. Vestibulum ac metus tempor sem mattis ornare at eget massa. Sed ut aliquam lorem.