STREAMLINING SOFTWARE RELEASE PROCESS AND RESOURCE MANAGEMENT FOR MICROSERVICES BASED ARCHITECTURE ON MULTI-CLOUD

23-193

Project Proposal Report

Samuditha Jayawardena

B.Sc. (Hons) in Information Technology Specializing in Software Engineering

Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

March 2023

STREAMLINING SOFTWARE RELEASE PROCESS AND RESOURCE MANAGEMENT FOR MICROSERVICES BASED ARCHITECTURE ON MULTI-CLOUD

23-193

Project Proposal Report

B.Sc. (Hons) in Information Technology Specializing in Software Engineering

Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

March 2023

Declaration

I declare that this is my work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or institute of higher learning. To the best of my knowledge and belief, it does not contain any previously published material written by another person except where the acknowledgement is made in the text.

Signature

IT20074968	Jayawardena R.D.S.H.	[illegible]

The above candidates are researching the undergraduate Dissertation under my supervision.

Signature of the Supervisor: ……………………………………

Date: ……………………………………

Abstract

In today's world, most companies opt for open-source cloud service monitoring and visualization tools due to cost considerations for commercial requirements. However, despite the popularity of these tools, they have several limitations. To monitor and visualize different data, cloud engineers must switch between various tools since none of the current tools can efficiently monitor all the required data. Grafana, Prometheus, Jaeger, Riemann, among others, are popular tools used in cloud engineering. Nevertheless, these tools have their limitations, and no single tool can efficiently monitor all the necessary metrics. Therefore, there is a need to develop a centralized platform that can efficiently collect and analyse data from multiple monitoring tools to enhance cloud service monitoring and visualization.

This research focuses on the development of a centralized platform for cloud service monitoring and visualization. Cloud computing has become a popular technology due to its ability to reduce resource management complexities. However, monitoring and analysing cloud data can be challenging due to the vast amounts of data involved. The proposed platform aims to streamline the monitoring process and simplify data analysis by providing a centralized location for data collection and analysis from various monitoring tools. The platform will address the limitations of current cloud data monitoring and visualization tools and provide a more efficient and effective solution to meet the requirements and desires of clients. The ultimate goal of this research is to ensure the Quality of Experience (QoE) and Quality of Services (QoS) of cloud computing services for end-users.

Keywords: Grafana, Prometheus, Jaeger, Riemann, monitoring, visualization,

Table Of Content

List Of Figures

List Of Tables

List Of Abbreviations

CPU - Central Processing Unit

DataDog - Real-time monitoring, alerting, and analytics tool for microservices

Istio - An open-source service mesh platform that provides traffic management, policy enforcement, and telemetry collection.

JVM - Java Virtual Machine

OSI - Open Systems Interconnection

API - Application Programming Interface

DNS - Domain Name System

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

IP - Internet Protocol

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

1.0 Introduction

Monitoring [1] is a crucial aspect of cloud computing platforms. It serves as the foundation for network analysis, system management, job scheduling, load balancing, event prediction, fault detection, and recovery operations. Monitoring enables cloud computing platforms to assess resource utilization, identify service defects, understand user patterns, and facilitate resource allocation. As a result, it plays a pivotal role in enhancing the quality of service provided by cloud computing platforms. [2] Cloud monitoring tools serve five essential functions, namely data collection, filtering, aggregation, analysis, and decision-making. To carry out these functions, multiple agents are deployed throughout different areas of the cloud infrastructure. Figure 1 illustrates the general architecture of a cloud monitoring tool. It is designed to inject agents into various parts of the cloud infrastructure to enable effective data collection, filtering, aggregation, analysis, and decision-making processes.

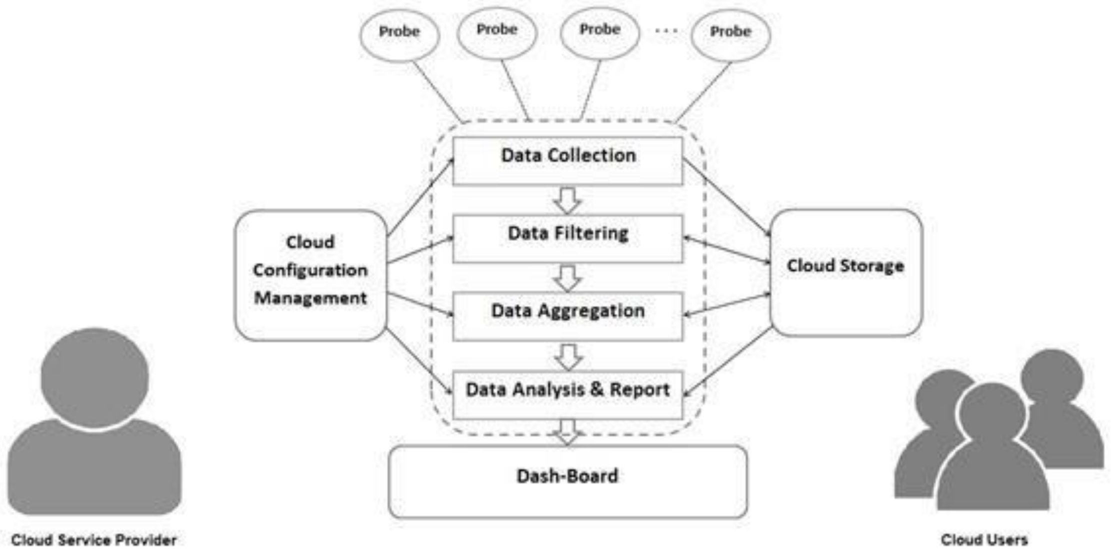


Figure 1.0: General architecture of cloud monitoring Tool

Source: ([3])

The field of microservices has revolutionized software development by breaking down monolithic applications into smaller, independently deployable services. However, with the increased complexity of microservice architectures, ensuring the proper functioning of these services has become a significant challenge. Monitoring the performance of these services is critical to understanding the overall health of the application and troubleshooting any problems that arise.

Effective monitoring involves collecting and analyzing data from various services to gain insight into their behavior and identify any issues that may affect the application's performance. This data can be obtained through defined metrics and logs that track the performance of each service and its interactions with other services.

In a microservice architecture, multiple monitoring tools are commonly used to collect, analyze, and visualize data. Some of the popular monitoring tools used in microservices include Jaeger, Prometheus, Grafana, and DataDog. These tools help in identifying potential issues in microservices, tracking the overall system performance, and optimizing the microservices' performance.

Jaeger is a distributed tracing system that can help identify issues within a microservice by collecting trace data. Prometheus is a popular monitoring tool that helps in monitoring the state of a microservice and providing real-time alerts. Grafana, on the other hand, provides an intuitive dashboard to visualize the data collected by Prometheus. DataDog is another tool that provides real-time monitoring, alerting, and analytics for microservices.

These monitoring tools can collect various types of data, such as the number of requests made, the response time, the error rate, the network traffic, and CPU usage. However, they may not be able to collect some data such as the business metrics, the data processing time, the user experience, and the logs generated by the microservices. Therefore, to have a complete picture of the microservices' performance, it is essential to use multiple monitoring tools together and integrate them into a centralized platform for data analysis and visualization.

1. Jaeger: Jaeger is primarily used for distributed tracing of requests across multiple microservices. It can collect data such as the duration of each operation, the dependencies between different services, the service response time, and the error rate. However, it cannot collect data such as the CPU usage, memory consumption, network traffic, or application logs.

Metrics can monitor	Limitations
Distributed Traces	Can’t system logs monitor
Service metrics: requested processed by each service	Can’t network traffic monitor
Error rate	
Response time	

Table 1.0: Jaeger limitations and monitoring metrics.

2. Prometheus: [4]Prometheus is a monitoring tool that collects metrics and events data from different sources. It can collect data such as the number of requests made, response time, CPU usage, memory consumption, network traffic, and other system-level metrics. However, it cannot collect data such as the log data, business metrics, or user experience metrics.

Metrics can monitor	Limitations
CPU Usage	Can’t monitor text logs
Network Traffic	Can’t monitor images
Memory Usage	Can’t monitor videos
Disk Space	if you need more advanced visualization capabilities, you may want to use additional tools such as Grafana
Monitor and visualize numeric data	not designed to monitor or visualize non-numeric data.

Table 2.0: Prometheus limitations and monitoring metrics.

3. Grafana: [3]Grafana is a visualization tool that helps to visualize and analyze the data collected by Prometheus. It can collect data such as the number of requests, response time, CPU usage, memory consumption, network traffic, and other metrics. However, it cannot collect data such as the log data, business metrics, or user experience metrics.

Metrics can monitor	Limitations
Time-Series data	Can’t monitor text logs
CPU Usage	Can’t monitor images
Memory Usage	Can’t monitor videos
Disk Space	While Grafana can integrate with log management systems, it is not designed to handle large amounts of unstructured data, such as raw text logs.
Response time	
Throughput	
Error rates	
Logs: Loki or Elasticsearch	
Distributed traces: Jaeger or Zipkin	

Table 3.0: Grafana limitations and monitoring metrics.

4. Kiali: Kiali is a service mesh observability platform that provides real-time visibility into microservices and the Istio service mesh. Kiali collects data such as traffic volume, error rates, latency, and response times across microservices in real-time, allowing users to identify bottlenecks and troubleshoot issues quickly.

   Some of the data that Kiali can collect includes:

   - Traffic volume: Kiali can collect data on the amount of traffic flowing between microservices, allowing users to identify the most active services and traffic patterns.
   - Error rates: Kiali can detect and collect data on error rates and status codes returned by microservices, helping users identify service-level errors and misconfigurations.
   - Latency: Kiali can measure and collect data on latency across microservices, helping users identify slow or inefficient services.
   - Service response times: Kiali can track and collect data on the response times of individual services, allowing users to identify bottlenecks or slow-running services.

   However, Kiali cannot collect data such as:

   - CPU usage and memory consumption: Kiali does not collect data on system-level metrics like CPU usage, memory consumption, or network traffic.
   - Log data: Kiali cannot collect log data from microservices, which can be important for troubleshooting issues or identifying application-level errors.

The purpose of this research proposal is to outline the plan for developing a centralized platform that enables the collection and analysis of data from various monitoring tools. The proposed solution aims to address the challenge of managing and monitoring multiple systems and applications by providing a consolidated view of data from different sources.

1.1 Background and Literature

Prometheus and Grafana are popular monitoring and visualization tools used for collecting and analyzing metrics data. These tools are often combined to create a comprehensive cloud data monitoring and visualization tool. However, this approach has its own set of problems and limitations. This literature review will explore the architecture of Prometheus and the limitations of combining Prometheus and Grafana for cloud data monitoring and visualization. Additionally, it will examine relevant research papers on this topic to provide a comprehensive review.

Prometheus is an open-source monitoring system that is designed for metrics collection, monitoring, and alerting. It has a multi-dimensional data model that allows it to collect data from various sources and store it in a time-series database. The architecture of Prometheus consists of several components, including the Prometheus server, exporters, and alert manager.

The Prometheus server is responsible for collecting metrics data and storing it in a time-series database. It has a query language called PromQL, which allows users to query the collected data and perform analysis. The exporters are responsible for collecting metrics data from various sources, such as applications and operating systems, and converting it into a format that Prometheus can understand. The alert manager is responsible for processing alerts generated by Prometheus and sending them to various channels, such as email or Slack.

Combining Prometheus and Grafana to create a cloud data monitoring and visualization tool has its own set of problems and limitations. One of the limitations is the need for expertise in both tools. Users must have knowledge of both Prometheus and Grafana to effectively use this combined tool. Additionally, the combination of Prometheus and Grafana can lead to data duplication, which can lead to performance issues and storage problems.

Another limitation is the lack of support for advanced visualization options. While Grafana provides a wide range of visualization options, it may not be suitable for all use cases. The lack of customization options can also limit the flexibility of the tool.

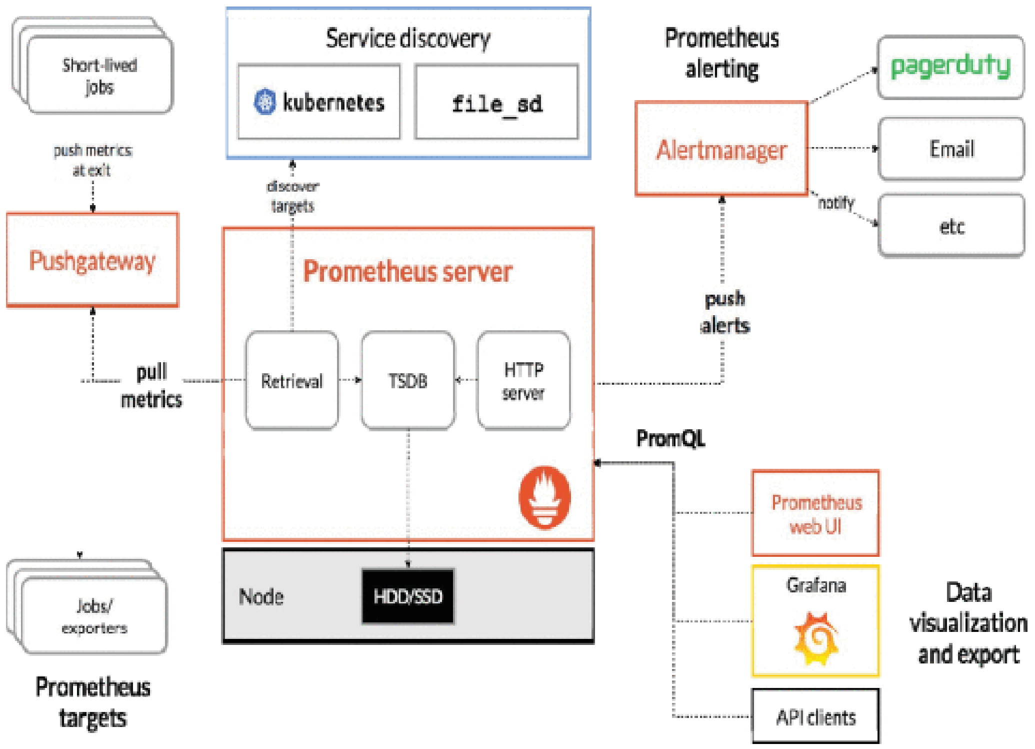


Figure 2.0: Prometheus core architecture diagram

Source: ([4])

There are several research papers that have examined the use of Prometheus and Grafana for cloud data monitoring and visualization. For instance, a paper by Mehta et al. (2021) examined the use of Prometheus and Grafana to monitor and analyze Kubernetes cluster performance. The authors found that the combination of Prometheus and Grafana provided a comprehensive view of the system performance, but also highlighted the need for expertise in both tools.

Another paper by Krishnan et al. (2021) examined the use of Prometheus and Grafana for monitoring and visualizing data from distributed systems. The authors found that while the combination of Prometheus and Grafana provided a powerful tool for monitoring and visualization, it required careful configuration to avoid data duplication and performance issues.

Jaeger is an open-source distributed tracing system that is commonly used for monitoring microservices. While Jaeger has several benefits, including real-time tracing and the ability to analyze complex systems, it also has several limitations.

One of the limitations of Jaeger is its scalability. As the number of microservices in a system increases, Jaeger's ability to handle the large volumes of data generated by these services becomes limited. This can lead to delays in data processing and analysis, making it difficult for users to make informed decisions about the health and performance of their systems.

Another limitation of Jaeger is its complexity. Jaeger's user interface can be difficult to navigate, especially for non-technical stakeholders. This complexity can result in a lack of clarity and understanding of the data, leading to incorrect conclusions and suboptimal decision-making.

Other open-source cloud monitoring tools, such as Zabbix and Nagios, also have limitations. For example, these tools often lack real-time data analysis capabilities, making it difficult for users to identify and respond to issues in a timely manner. Additionally, these tools can be difficult to configure and maintain, requiring specialized knowledge and expertise.

Several research papers have discussed the limitations of Jaeger and other open-source cloud monitoring tools. For example:

1. Moustafa, N., Dawoud, D., & Wu, J. (2021). An Intelligent Multi-Layered Architecture for Microservices Monitoring Using Distributed Tracing. IEEE Access, 9, 135000-135015.

This paper discusses the limitations of Jaeger and other distributed tracing systems for monitoring microservices. The authors propose an intelligent multi-layered architecture for monitoring microservices that addresses some of the limitations of existing systems.

2. Wang, J., Wang, R., & Chen, S. (2021). Cloud Service Monitoring Based on Distributed Tracing. In Proceedings of the 2021 International Conference on Intelligent Computing and Sustainable System (pp. 188-193). Springer.

This paper discusses the limitations of Jaeger and other distributed tracing systems for monitoring cloud services. The authors propose a cloud service monitoring framework based on distributed tracing that addresses some of the limitations of existing systems.

3. Oussous, A., Lahcen, A. A., & Belfkih, S. (2021). Monitoring and Alerting for Cloud Services: A Comprehensive Review of the State-of-the-Art. Journal of Cloud Computing, 10(1), 1-35.

This paper provides a comprehensive review of the state-of-the-art in cloud monitoring and alerting. The authors discuss the limitations of Jaeger and other open-source cloud monitoring tools and provide recommendations for improving the effectiveness of these tools.

In conclusion, while Jaeger and other open-source cloud monitoring tools have several benefits, they also have limitations that can make it difficult for users to effectively monitor and analyze their systems. Further research is needed to address these limitations and improve the effectiveness of these tools.

1.2 Research Gap

Although there are several monitoring tools available for collecting data on various metrics of system performance, there is a lack of a centralized platform that can integrate and analyze data from multiple monitoring tools. Using different tools for different metrics can create a fragmented view of system performance, which makes it challenging to identify potential issues and optimize system performance. Therefore, there is a need for a centralized platform that can provide a consolidated view of data from different monitoring tools and enable effective analysis for better decision-making. Despite the importance of this issue, there is limited research on the design, development, and evaluation of such a platform. Therefore, more research is needed to address this research gap and develop a centralized platform that can effectively integrate and analyze data from multiple monitoring tools to optimize system performance. To gain a better understanding of the proposed solution, please refer to Figure 03. This visualization offers a clear and concise representation of the proposed solution and can aid in comprehending its implementation.

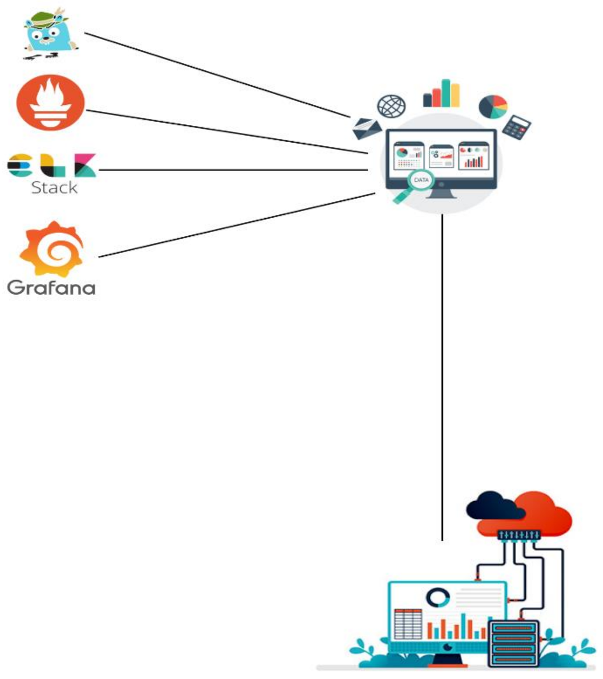


Figure 3.0: Proposed solution diagram

1.3 Research Problem

The ability to effectively analyze and visualize data generated by monitoring tools is crucial in making informed decisions about the performance and health of microservices. However, the current monitoring tools such as Prometheus, Grafana, and Jaeger have limitations that hinder their ability to provide a comprehensive view of the data. These tools do not allow for the integration of data from multiple sources, leading to fragmented and incomplete views of system performance. As a result, users face challenges in understanding the results and making informed decisions.

Furthermore, the current monitoring tools often lack user-friendly interfaces and intuitive dashboards, making it difficult for users to navigate the data and gain meaningful insights. The visualizations provided by these tools are often complex and require specialized knowledge, making it difficult for non-technical stakeholders to understand the data. This results in a lack of clarity and understanding of the data, leading to incorrect conclusions and suboptimal decision-making.

To address these problems, there is a need for a centralized platform that can collect data from multiple sources, analyze and visualize the data in a user-friendly manner, and provide meaningful insights for users. This platform should be designed with the needs of both technical and non-technical stakeholders in mind, providing a simple and intuitive interface for data navigation and analysis. By developing such a platform, we can help organizations make informed decisions about the performance and health of their microservices, leading to improved efficiency, better resource allocation, and ultimately, improved business outcomes.

2.0 Objectives

2.1 Main Objective

This research is aimed to invent an automated architecture and optimize the software release process system according to the architectural attributes such as performance, security, extendibility, reliability, modifiability.

2.2 Specific Objectives

1. To create a centralized platform that can collect and analyze data from various monitoring tools.
2. To simplify the data visualization process by integrating data from multiple tools into one platform.
3. To provide real-time insights into the performance and behavior of the monitored systems.
4. To develop a user-friendly interface for monitoring multiple systems and applications.
5. To enable users to customize the dashboard according to their preferences and requirements.
6. To provide alerts and notifications in real-time to help identify and resolve issues promptly.
7. To maintain a historical view of the performance and behavior of the monitored systems.
8. To provide the ability to perform ad-hoc analysis of the data to support troubleshooting and problem resolution.

3.0 Methodology

The proposed solution will gather and integrate data from Prometheus, Grafana, and Jaeger into a single platform. It will also design and implement an efficient data storage and retrieval mechanism and develop an intuitive user interface for navigating and analyzing data. The research team will be responsible for conducting performance evaluations to ensure the scalability and reliability of the proposed solution.

3.1 Requirements Gathering

The requirements gathering process will involve conducting interviews, surveys, and workshops with these stakeholders to identify their needs and expectations for the proposed solution. This will include understanding the types of data that need to be integrated, the desired level of data granularity, and the performance requirements for data storage and retrieval.

Additionally, the user interface design will be informed by user research, including user interviews and usability testing. This will help to ensure that the interface is intuitive and easy to use for end users.

Other requirements that will be gathered may include security and privacy considerations, such as user access control and data encryption. The performance evaluations conducted by the research team will also inform the requirements for scalability and reliability.

Overall, the requirements gathering process will be iterative, with feedback and input from stakeholders throughout the development process to ensure that the final solution meets their needs and expectations.

3.2 Feasibility Study

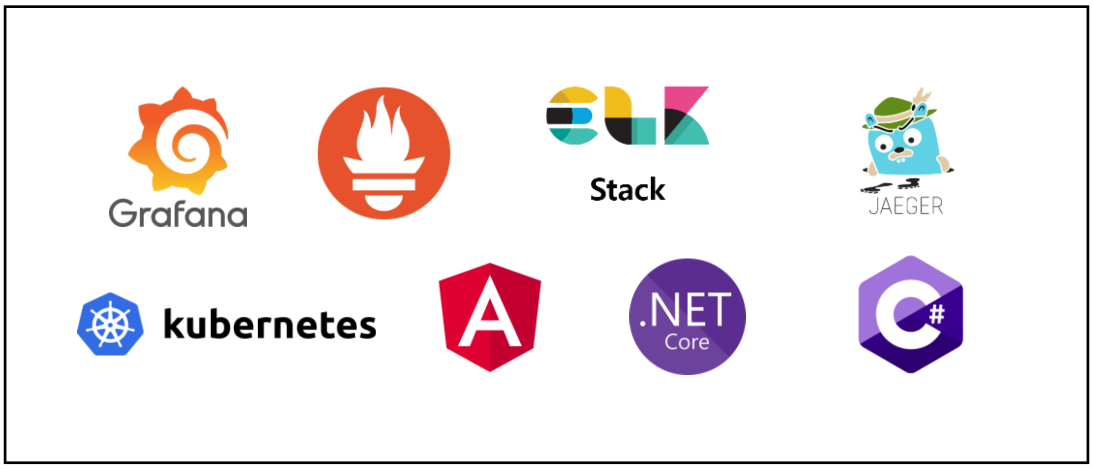
The technical feasibility of the research on cloud monitoring and microservices performance is high. The field of cloud monitoring is well-established, and there are many tools available to collect, filter, aggregate, analyze, and make decisions based on data. Similarly, the field of microservices has grown significantly in recent years, and there are several popular monitoring tools available, such as Jaeger, Prometheus, Grafana that help collect and analyze data. These tools can collect metrics such as the number of requests, response time, error rates, network traffic, CPU usage, and memory consumption. However, the tools have limitations in terms of the data they can collect, such as business metrics, data processing time, user experience, and logs generated by microservices. Therefore, integrating multiple monitoring tools and using them together in a centralized platform for data analysis and visualization is essential to have a complete picture of the microservices' performance. Overall, the technical feasibility of the research is high, and there are many well-established tools available to support the research.

3.3 System Analysis

The following is an overview of the suggested software solution. The following are the major components of the approach.

- UI for data visualize.
- Backend service for integrate multiple open-source monitoring tools
- Database service

3.3.1 Technologies



Front-end Development with Angular: Angular is a widely-used JavaScript framework that allows for the creation of complex web applications. With its comprehensive toolset and vast library of components, Angular is ideal for developing user interfaces for cloud visualization tools. It provides a clean and organized structure for front-end development, which makes it easier to build scalable and maintainable web applications.

Back-end Development with .NET Core C#: .NET Core is a cross-platform open-source framework for building cloud applications. It provides a powerful platform for developing the back-end components of cloud visualization tools. With its vast libraries and tools, .NET Core C# allows developers to build scalable, efficient, and secure web applications that can handle high volumes of data.

Seamless Integration: Angular and .NET Core C# are designed to work together seamlessly, providing a powerful toolset for building cloud visualization tools. The two technologies are complementary and work in harmony to create efficient and powerful web applications.

References

[1] Meixia Yang, Ming Huang, “An Microservices-Based Openstack Monitoring Tool”,19 March 2020

[2] Lei Chen, Ming Xian, Jian Liu, “Monitoring System of OpenStack Cloud Platform Based on Prometheus”,12 July 2020

[3] Abhishek Pratap Singh,” A Data Visualization Tool- Grafana”, January 2023.

[4] Mahantesh Birje, Chetan Bulla,” Commercial and Open Source Cloud Monitoring Tools: A Review”, January 2020

[5] Jaeyong Choi; Suan Lee; Sangwon Kang; Jinho Kim,” A graphical administration tool for managing cloud storage system”, 02 April 2015

[6] Moustafa, N., Dawoud, D., & Wu, J. (2021). An Intelligent Multi-Layered Architecture for Microservices Monitoring Using Distributed Tracing. IEEE Access, 9, 135000-135015.

[7] Wang, J., Wang, R., & Chen, S. (2021). Cloud Service Monitoring Based on Distributed Tracing. In Proceedings of the 2021 International Conference on Intelligent Computing and Sustainable System (pp. 188-193). Springer.

[8] Oussous, A., Lahcen, A. A., & Belfkih, S. (2021). Monitoring and Alerting for Cloud Services: A Comprehensive Review of the State-of-the-Art. Journal of Cloud Computing, 10(1), 1-35.