Microservice Architecture in E-commerce Platforms using Docker and Kubernetes

Abdiaziz Abdullahi Aden
School Of Engineering & Technology
Sharda University
abdiaziiz1856@gmail.com

Dr.Nishant Gupta
Associate Professor
School Of Engineering & Technology
Sharda University
nishantlira@gmail.com

Dr. Priyanka Tyagi
Assistant Professor
School Of Engineering & Technology
Sharda University
Priyanka.tyagi@sharda.ac.in

***Abstract*— The microservices architecture transforms software development and deployment by increasing modularity, scalability, and flexibility. This technique divides apps into smaller, independent services that communicate over HTTP APIs. Docker plays an important role in containerizing these services, guaranteeing consistent operation across several settings. Docker containers are lightweight, use the host kernel, and start quicker than typical virtual machines, making them perfect for microservices. Kubernetes, Google's open-source technology, enhances microservices by automating container deployment, scaling, and administration across host clusters. This combination creates a stable environment for delivering microservices, allowing for continuous integration and deployment (CI/CD). This article investigates the synergistic usage of microservices, Docker, and Kubernetes, showing their combined efficiency in developing, deploying, and scaling applications across many environments.**

Keywords*— *microservice, architecture, Monolithic Architecture, Kubernetes, Docker

I. INTRODUCTION

Microservice architectures are designed to enable the deployment and scaling of small services that can be implemented independently of one another and may use a variety of middleware stacks. These architectures are intended to address the drawbacks of traditional monolithic architectures, where all an application's logic and data are managed in a single deployable[1]. In recent years, microservice architecture has been gaining more and more attention due to its benefits compared to traditional, monolithic systems. Microservice architecture breaks down systems into smaller, loosely linked parts called Microservices. Each Microservice is an independent part that takes care of a particular business function [2].

A microservices architecture brings significant benefits, including the ability to use different technologies across services, enhancing system resilience by isolating failures to single components and offering more efficient scalability by targeting specific services rather than the entire system. It simplifies deployment processes with independent service updates and aligns with organizational structures to streamline team workflows. Additionally, it supports service composability and facilitates easier service replacement, according to Newman's insights in "Building Microservices"[3].

This paper examines the adoption of Microservice Architecture in e-commerce platforms, with a focus on leveraging Docker and Kubernetes for this transformation. It discusses how these technologies address challenges like scalability, deployment, and resilience that are inherent in traditional monolithic systems by segmenting complex applications into smaller, standalone services.

II. PROBLEM STATEMENT

In the e-commerce platforms, there is a critical need to overcome the limitations posed by traditional monolithic architectures, including challenges related to scalability, deployment agility, service isolation, and fault tolerance. The goal is to design and implement a microservice architecture that leverages Docker and Kubernetes to address these issues. This involves creating a scalable, flexible, and resilient system that supports continuous deployment and operational efficiency, while also providing a framework that reduces technology lock-in and simplifies the management and orchestration of numerous microservices. The specific problem is how to effectively utilize Docker for containerization and Kubernetes for orchestration to achieve these objectives in an e-commerce setting.

III. LITERATURE REVIEW

This manuscript part examines the literature on microservices and compares their efficiency to other architectural models such as monolithic and service-oriented architectures. It also contains views from many researchers on the fundamentals of microservice architecture. Singh and Peddoju's study compared the performance of a typical monolithic program to one built on microservices, doing trials using 2000 threads. Their findings revealed that microservices perform better in cases with a large number of requests [4]. In contrast, an IBM team examined how different architectures utilize resources under various operational settings and hardware configurations, and discovered that monolithic apps typically outperformed microservices, contradicting Singh and Peddoju's findings.[5] This contradiction merits more consideration in this study. Furthermore, Mark Richards examines these notions by comparing microservices to Service Oriented Architectures in his publications. emphasizing variations in service dynamics and architectural strengths. Finally, Villamizar and his colleagues studied the financial and operational implications of deploying web solutions in the cloud using various architectures such as microservices, classic monolithic, and AWS Lambda. Their investigation found that microservices tend to delay down replies since each service request requires more processing at the gateway.[6]

A. *Microservices*

Applications created using microservices thrive at horizontal scalability, both technically and organizationally. This enables the construction of smaller, more agile development teams. Initiatives to include these techniques in flexible business process management have been applied effectively in the sector. Decomposing huge programs into individual microservices also gives agile teams more autonomy, which improves the scalability of agile methods.[7]

Teams may work independently on different microservices, building specialized functionality without affecting others, as long as the inter-service agreements stay unchanged.[8] This autonomy reduces the need for cross-team cooperation, which is frequently a bottleneck in large-scale software projects. Microservice designs provide enhanced resilience since their components are loosely connected. The failure of one microservice does not bring down the entire system, enabling the operational components to continue serving requests. Critical functions can be strengthened by putting them in redundantly built settings to provide greater dependability.[9]

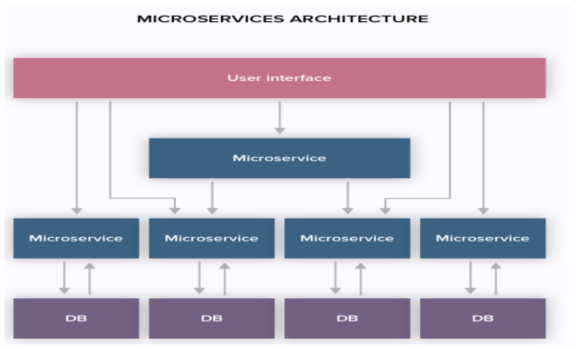

Figure 1: Microservices Architecture

This investigation of performance trade-offs in microservice architectures is a significant focus of our study. Containerization and virtualization technologies such as Docker and Kubernetes are recommended for successfully implementing a microservices architecture. These technologies make it easier to install and manage microservices in cloud-based settings, assuring their efficiency and reliability.[10]

B. *Docker*

Docker, an open-source platform, makes it easier to create and distribute programs by packaging them with all of their dependencies into containers. These containers run in isolation on the operating system's kernel, adding a layer that may affect performance. Although container technologies have been around for more than a decade, Docker stands out with novel capabilities that were not accessible in earlier technologies. It offers simple container administration, enabling developers to pack programs into lightweight containers for convenient operation anywhere, supports a larger density of virtual environments on the same hardware, and interfaces seamlessly with third-party deployment and management tools. Docker containers are also suitable for cloud settings.[11]

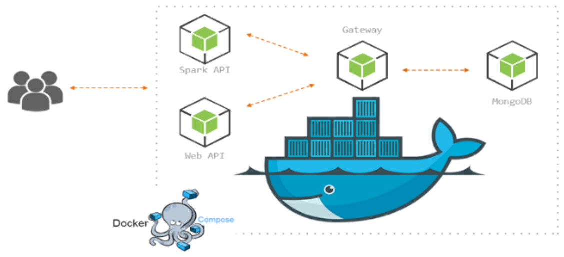

Figure 2: Docker Containers

C. *Kubernetes.*

Kubernetes, a widely adopted open-source system, facilitates the automatic handling of containerized applications, enhancing scalability and management. Esteemed entities such as IBM and the U.S. Department of Defense leverage this technology for efficient container oversight and rapid deployment cycles, dramatically shortening the time to release software updates.[12]

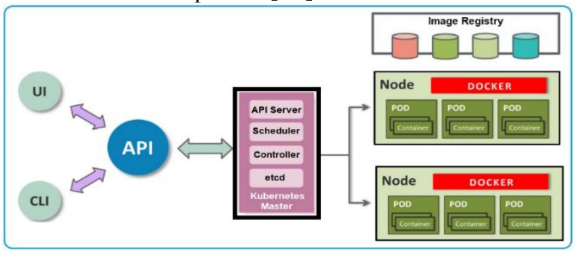

Figure 3: Kubernetes Cluster

IV. LITERATURE SURVEY TABLE

Sr. NO	***Authors***	***Techniques***	***Outcomes***	***Limitations***
1	Savi **[3]**	Uses a scalable microservice-based architecture for DMTF profiles	Demonstrates enhanced scalability and flexibility in IT systems	Specific challenges and limitations were not detailed in the text
2	Taibi, Davide Lenarduzzi, Valentina Pahl, Claus **[8]**	Systematic mapping study to identify microservice architecture patterns	Catalogue of patterns for orchestration, deployment, and data storage	Lack of empirical validation and benchmarks to compare patterns
3	Sanctis, Martina De Muccini, Henry Vaidhyanathan, Karthik **[9]**	Designs a self-adaptive architecture for microservice-based IoT systems Uses machine learning	Proposes an architecture to handle adaptations at different levels in microservice-based IoT systems	Lacks implementation details and evaluation
4	Perera, K. J.P.G. Perera, I. **[5]**	Input wizard to gather system requirements Data processor to map requirements to models Architecture generator algorithm to identify components Visual representation of architecture	Tool to automatically generate microservice/serverless architecture from requirements	Limited evaluation on a few sample applications Lacks knowledge base for refinement
5	Ramu, Vivek Basavegowda **[2]**	enhance the performance of microservices-based systems, focusing on efficient inter-service communication, and scalability	Provide practical insights and recommendations from literature and industry best practices.	Absence of clarity on the depth of investigation into elements like inter-service communication, service discovery, data management, fault tolerance, and scalability.
6	Raffin, Tim Mayr, Andreas Fuchs, Jonathan Baader, Marcel Morello, Andreas Kuhl, Alexander Franke, Jorg **[7]**	Using a software architecture where data acquisition is handled by small, independent services.	Flexible and reliable data acquisition, Low latency and high data integrity, Adaptability to different communication protocols	Limitations of existing data acquisition systems and discarded feature-rich data
7	Hasselbring, Wilhelm Steinacker, Guido **[1]**	Microservices for Scalability and Fault Tolerance	Enhanced non-functional attributes like scalability and fault tolerance, allowing applications to tolerate the failure of individual services and utilize real-time monitoring.	Necessity to design applications to detect failures quickly and restore services.

8	I Asrowardi, S D Putra, E Subyantoro **[4]**	Microservice Architecture	Microservices architecture demonstrates better performance in scripting and painting compared to the monolithic architecture	The main challenge highlighted is the migration from monolithic to microservices architecture, including deployment artifacts and details.
9	Isak Shabani, Endrit Mëziu, Blend Berisha, Tonit Biba **[10]**	Microservices Architecture Decomposition	Decomposed a monolithic e-commerce application into microservices ,demonstrating the application of microservices architecture through a .NET framework MVC pattern.	The paper presents a successful decomposition and comparison but does not extensively cover the challenges in maintaining and monitoring a distributed microservices system in operation.
10	L.D.S.B Weerasinghe, I Perera **[6]**	Experimental Methodology on Cloud Platforms	Conducted an experimental analysis of inter-service communication mechanisms on AWS cloud platform, using Java and Spring Boot for microservice development	The research focuses solely on performance aspects and does not explore other critical factors such as security, ease of implementation, and compatibility with different cloud environments or programming languages, which could influence the choice of inter-service communication protocol.
11 **[12]**	Shazibul Islam Shamim et al.	Multi-vocal Literature Review (MLR)	Identified benefits, challenges, and research topics in Kubernetes usage.	Limited to the collected peer-reviewed publications and Internet artifacts; potential biases in selection and ratings.
12	Babak Bashari Rad, Harrison John Bhatti, Mohammad Ahmadi **[11]**	Docker and Containerization Performance Analysis	Demonstrated Docker's superior performance over traditional virtual machines in terms of resource utilization	Limited testing environments, possible biases in performance benchmarks due to specific configurations used.

CONCLUSION

Finally, microservice architectures have developed as a transformational method for software development, providing a break from the constraints of old monolithic systems. They divide big programs into smaller, self-contained microservices that manage certain business operations on their own, resulting in enhanced scalability and fault tolerance. The use of HTTP for communications simplifies the scaling process and allows services to be deployed on numerous servers, making it easier to accommodate rising demand. Microservices can be thought of as a particular implementation strategy within the larger framework of service-oriented architecture (SOA). Although there is no one owner or originator, there is agreement among industry participants who have contributed to the evolution and maturation of this architectural style. Its success and profitability for organizations such as Amazon and Microsoft may be linked to its multi-level working paradigm, which provides a strong framework for creating flexible, self-provisioning software systems. As the software environment evolves, microservices provide a viable path to increased agility and scalability in application development.

REFERENCES

[1] W. Hasselbring and G. Steinacker, “Microservice architectures for scalability, agility and reliability in e-commerce,” in Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings, Institute of Electrical and Electronics Engineers Inc., Jun. 2017, pp. 243–246. doi: 10.1109/ICSAW.2017.11.
[2] V. B. Ramu, “Performance Impact of Microservices Architecture,” Rev. Contemp. Sci. Acad. Stud., vol. 3, no. 6, Jun. 2023, doi: 10.55454/rcsas.3.06.2023.010.
[3] siva, “Scalable microservice based architecture for enabling DMTF profiles,” in Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015, 2015, pp. 428–432. doi: 10.1109/CNSM.2015.7367395.
[4] I. Asrowardi, S. D. Putra, and E. Subyantoro, “Designing microservice architectures for scalability and reliability in e-commerce,” J. Phys. Conf. Ser., vol. 1450, no. 1, 2020, doi: 10.1088/1742-6596/1450/1/012077.
[5] K. J. P. G. Perera and I. Perera, “TheArchitect: A Serverless-Microservices Based High-level Architecture Generation Tool,” Proc. - 17th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2018, pp. 204–210, 2018, doi: 10.1109/ICIS.2018.8466390.
[6] L. D. S. B. Weerasinghe and I. Perera, “Evaluating the Inter-Service Communication on Microservice Architecture,” 7th Int. Conf. Inf. Technol. Res. Digit. Resil. Reinvention, ICITR 2022 - Proc., pp. 1–6, 2022, doi: 10.1109/ICITR57877.2022.9992918.

[7] T. Raffin et al., “A Microservice-Based Architecture for Flexible Data Acquisition at the Edge in the Context of Hairpin Stator Production,” 2021 11th Int. Electr. Drives Prod. Conf. EDPC 2021 - Proc., 2021, doi: 10.1109/EDPC53547.2021.9684194.
[8] D. Taibi, V. Lenarduzzi, and C. Pahl, “Architectural patterns for microservices: A systematic mapping study,” CLOSER 2018 - Proc. 8th Int. Conf. Cloud Comput. Serv. Sci., vol. 2018-Janua, pp. 221–232, 2018, doi: 10.5220/0006798302210232.
[9] M. De Sanctis, H. Muccini, and K. Vaidhyanathan, “Data-driven Adaptation in Microservice-based IoT Architectures,” Proc. - 2020 IEEE Int. Conf. Softw. Archit. Companion, ICSA-C 2020, pp. 59–62, 2020, doi: 10.1109/ICSA-C50368.2020.00019.
[10] I. Shabani, E. Mëziu, B. Berisha, and T. Biba, “Design of Modern Distributed Systems based on Microservices Architecture,” Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 2, pp. 153–159, 2021, doi: 10.14569/IJACSA.2021.0120220.
[11] B. B. Rad, H. J. Bhatti, and M. Ahmadi, “An Introduction to Docker and Analysis of its Performance An Introduction to Docker and Analysis of its Performance,” no. February 2022, 2017.
[12] S. Islam, S. Jonathan, A. Gibson, and P. Morrison, “Benefits ,
Chllenges , and Research Topics : A Multi-vocal Literature Review of Kubernetes”.