International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Scalable Microservices Development using Java and AWS Lambda

Hareesh Kumar Rapolu

hareeshkumar.rapolu@gmail.com

Abstract

The research paper has explored the concept of scalable microservices. The study has analysed the different ways in which these microservices are developed with the help of AWS Lambda and Java platforms. Different businesses are using these services to improve the way in which they cater to their customers. The study has also identified a number of benefits of using scalable microservices. In today's world, developers gain comprehensive knowledge about these scalable microservices to create better applications.

Keywords: Scalable microservices, Java, AWS Lambda, Microservices development

I. INTRODUCTION

The approach to software development is rapidly changing. The developers now rely on a range of scalable microservices that make it a lot easier to craft a new application. This research paper will shed light on how scalable microservices are developed using both Java and AWS Lambda. The study will further elaborate on a few benefits that are experienced from developing these microservices.

II. UNDERSTANDING THE CONCEPT OF SCALABLE MICROSERVICES

Microservices can be described as a software development style where the developer breaks the software into multiple small and independent services. In this regard, it needs to be mentioned that each one of the microservices carries out a specific task. Despite being independent, they communicate with one another using the Application Programming Interface (API) that is designed by the developer. The business world has become extremely complex and ambiguous in nature. Therefore, microservices have become absolutely crucial for a particular business to maintain its IT services. The microservices architecture is a lot more flexible than the monolithic architecture. Within the conventional monolithic design, all the different processes that are inherent within an application are tightly coupled with one another. As a result, if that particular application experiences a rise in demand, its entire architecture has to be appropriately scaled. In such instances, the microservice architecture becomes a lot more practical where each service is loosely coupled and runs independently from one another[1]. Most importantly, the entire microservices architecture is extremely lightweight. Hence, it is extremely easy for a developer to scale their applications and services in order to cater to the increase in business demands. This is one of the major reasons why scalable microservices are used by different kinds of businesses all across the world. Therefore, it is quite evident that two particular characteristics of microservices help them to stand out in

the contemporary world. Primarily, each service within the entire microservices infrastructure is autonomous in nature[2]. Therefore, one singular service can be developed, deployed and scaled without impacting the function of other services. None of them share their inherent codes with one another. This is how they are extremely secure and reliable. The second important characteristic is that each microservice is specifically designed for solving a particular problem. This streamlined nature of each microservice helps to maintain larger applications. It needs to be mentioned that the availability of the microservices is measured by dividing total uptime by the summation of total uptime and downtime.

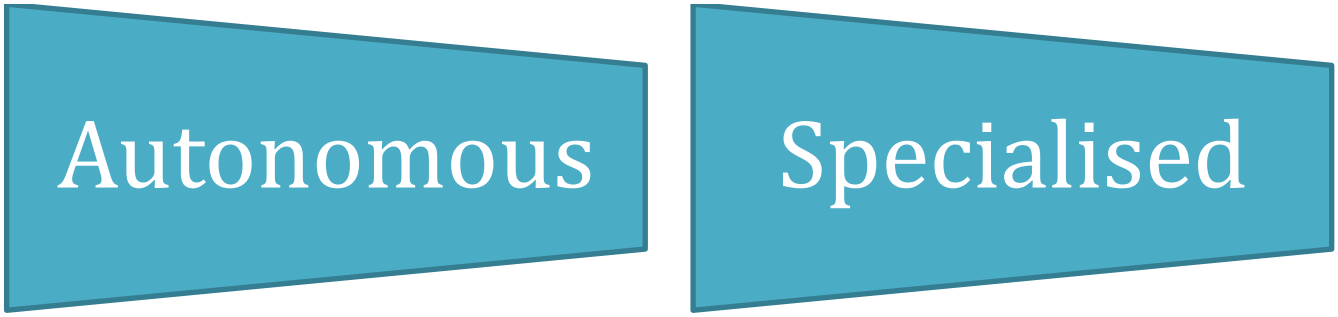

Figure 1: Chatacteristics of microservices

III. HOW SCALABLE MICROSERVICES ARE DEVELOPED USING JAVA AND AWS LAMBDA

Java

Sprint Boot: This is a widely used Java framework that offers a lot of tools and features for developing microservices. It also provides pre-configured templates which can be used to create a microservice. Most importantly, Sprint Boot is extremely compatible with multiple data sources.

Vert.x: Scalable and highly responsive microservices can be built using the Vert.x framework. It has support for both asynchronous and synchronous programming models. In addition, the concurrency capability of this framework can aid developers in building scalable microservices.

Micronaut: Micronaut is a relatively new framework of Java. It can be useful for developing serverless applications and scalable microservices. Some essential features of this framework include fast startup time, reduced memory footprint and support for cloud-native services like load balancing[3]. With the robust capabilities of Java, the developers can ensure that future-proof applications can be built which can adapt to changing business situations.

AWS Lambda

In AWS Lambda, each scalable microservice is defined as a Lambda function. In the next step, the developer needs to create API Gateway points for each microservice in order to trigger the previously created Lambda functions. After getting triggered by the API Gateway, each Lambda function runs the

microservice code and returns a response[4]. The API Gateway is instrumental in routing the HTTP requests that are made to trigger the Lambda functions. In this context, the cost of Lambda concurrency is calculated by dividing the incoming request rate by the execution duration per request.

IV. BENEFITS OF SCALABLE MICROSERVICES

Increased agility

This advantage is mainly achieved because each microservice acts as a small and independent team that is responsible for carrying out certain operations. Therefore, the developers can create them within a shorter span of time[5]. In this manner, the business benefits from the entire throughput that is produced by the microservices architecture.

Flexible scaling

Each microservice can be individually scaled because they operate independently from one another[6]. This helps the business to allocate the right amount of resources and manage its costs in an effective manner.

Easy deployment

Since the development cycle times are considerably lowered, it does not take a lot of time for the developers to roll back a certain idea and deploy a new one as required. Moreover, the cost of failure is very low. This is why the developers are able to experiment freely.

Technological freedom

The departments within a business or allowed to experiment with the microservices architecture. There is no one-size-fits-all approach for building and managing the microservices. The freedom to choose the best tool for operating within the architecture helps the workers to become more creative.

Reusable codes

Functions are an essential part of a coding language. Since a particular software is divided into small, well-defined sections, the IT team is able to use one function for different purposes. Therefore, a code that is written for a particular function can potentially become the building block of another feature within the microservices architecture.

Resilience

This is one of the most important benefits of building scalable microservices. In the scalable microservices architecture, the failure of one single service does not mean that the entire architecture needs to be restructured[7]. In this manner, the IT department of a business can keep on developing new applications even if one single service suddenly fails.

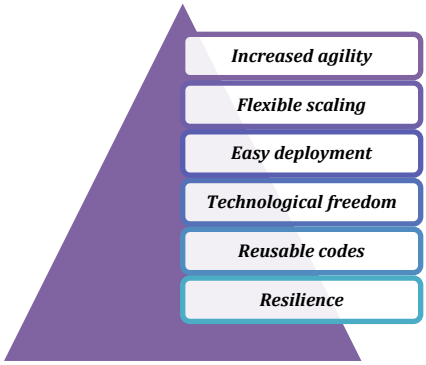

Figure 2: Benefits of scalable microservices

V. CONCLUSION

It can be concluded that the lightweight architecture of the scalable microservices can be built using the Java frameworks and AWS Lambda as well. These microservices offer better reliability and can be useful in building future-proof applications.

Abbreviations and acronyms

- API - Application Programming Interface
- IT - Information Technology
- AWS - Amazon Web Services

Units

- Response time - Milliseconds
- Memory allocation (MB)
- Network latency - Milliseconds
- Bandwidth - Megabits per second (Mbps)

Equations

- A=U/U+D
- Nmax=Q/Texec

REFERENCES

[1] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, "From monolithic systems to Microservices: An assessment framework," *Information and Software Technology*, vol. 137, Sep. 2021, doi: https://doi.org/10.1016 /j.infsof.2021.106600

[2] F. Freitas, A. Ferreira, and J. Cunha, "Refactoring Java Monoliths into Executable Microservice-Based Applications," *Proceedings of the 25th Brazilian Symposium on Programming Languages*, pp. 100–107, Oct. 2021, doi: https://doi.org/10.1145 /3475061.3475086

[3] G. BLINOWSKI, A. OJDOWSKA, and A. PRZYBYŁEK, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, Feb. 2022, doi: https://doi.org/10.1109/ACCESS.2022.3152803

[4] M. Villamizar *et al.*, "Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures," *Service Oriented Computing and Applications*, vol. 11, Apr. 2017, doi: https://doi.org/10.1007/s11761-017-0208-y

[5] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges," *Journal of Computer Information Systems*, Sep. 2018, doi: https://doi.org/10.1080/08874417.2018.1520056

[6] S. De Santis, L. Florez, D. V. Nguyen, and E. Rosa, *Evolve the Monolith to Microservices with Java and Node*. IBM Redbooks, Dec. 2016. Available: https://books.google.com/books?hl=en&lr=&id=oVmmDQAAQBAJ&oi=fnd&pg=PP1&dq=SCALABLE+MICROSERVICES+DEVELOPMENT+USING+JAVA&ots=Drct3IQhon&sig=n2iSe1TeeFDvIhKBFZtAOIJhEws

[7] S. Rabiu, C. H. Yong, and S. M. S. Mohamad, "A cloud-based container microservices: a review on load-balancing and auto-scaling issues," *International Journal of Data Science*, vol. 3, no. 2, Dec. 2022, Available: https://www.ijods.org/index.php/ds/article/download/45/38