MICROSERVICES DEVELOPMENT AND MANAGEMENT

Sayfullaev Shakhzod Oybek Ugli

Abstract. *This article is about microservices and their development and management.*

Key words. *Microservice, technical service, energy, software, architecture, program.*

Microservices architecture, increasingly adopted by companies such as Amazon, Netflix and PayPal, breaks software applications into smaller, independent units to increase agility and efficiency. This approach aligns well with agile business models and small, cross-functional teams, increasing maintainability, deployment, and scalability. This benefits cloud providers by promoting cloud-based platforms, reducing infrastructure needs. This evolution in software development is driving innovation, reducing costs and enabling global collaboration, as well as creating new security challenges. In general, microservices lead to faster market adaptation, more energy-efficient computing, and increased demand for skilled professionals in the field.

Microservices is a software architecture style that structures an application as a collection of loosely coupled, independently deployable services. Each microservice is designed to perform a specific business function, allowing us to develop, deploy, and extend it independently of other services within the application. Therefore, this approach differs from the traditional monolithic architecture, where all components are woven into one large code base.

The microservices development paradigm emerged as an evolutionary approach to building software applications as modular and independently deployable units. The main goal of adopting this paradigm is to divide business processes into smaller, autonomous parts called services. Each service has its own container, programming language, process, data storage and communication mechanisms. This approach provides software developers with a framework that minimizes implementation effort and offers a lightweight, flexible, and extensible way to build and run applications. In addition, MSA offers a number of advantages over monolithic software architectures, such as improved serviceability, deployability, testability, extensibility, integration, and flexibility.

Key features of microservices

1. Single responsibility: Each microservice focuses on a specific business opportunity or function.

2. Independence: Microservices can be developed, deployed and extended independently.

3. Autonomous: Each microservice is typically an independent entity with its own database and dedicated resources.

4. Communication: Microservices communicate with each other through well-defined APIs, often using HTTP/REST, gRPC, or message queues.

5. Decentralized data management: Each microservice manages its own data, promoting data encapsulation and autonomy.

6. Polyglot programming: We can develop different microservices using different programming languages and technologies, which are best suited for their specific tasks.

7. Fault Isolation: Failures in one microservice do not propagate to other services, which increases the overall resilience of the application.

1. Advantages of microservices

2. Scalability: Microservices allow individual components to scale independently of each other based on demand.

3. Flexibility: Developers can use different technologies and frameworks for different services.

4. Resilience: If one microservice fails, it does not mean that the entire system is down.

5. Faster development: Smaller teams can work on different services simultaneously, which speeds up development cycles.

6. Easier to implement: CI/CD pipelines (Continuous Integration and Continuous Deployment) we can implement more easily.

7. More Maintainable: Smaller, more focused codebases are easier to manage and understand.

8. Reusability: We can reuse commonly used microservices in different applications, reducing development time and effort.

Disadvantages

1. Complexity: The architecture introduces additional complexity in the areas of interservice communication, data consistency, and service discovery.

2. Operational costs: Powerful monitoring, logging and management tools are required to manage large number of services.

3. Data Management: Ensuring data consistency and managing distributed transactions can be difficult.

4. Latency: Communication between services over the network can cause delays.

5. Security: We need to protect each service separately, which increases the cost of security management

Microservices are a powerful architectural style, but they are not necessarily appropriate for every type of software. The decision to use microservices for reengineering should be based on specific criteria and the specific needs of the project. Thus, by following a systematic approach and considering potential problems, we can efficiently transform legacy software into a modern, microservices-based system. This provides long-term benefits in terms of scalability, flexibility and sustainability

It is worth noting that microservices have gained popularity not only among developers, but also among managers and project managers. The explanation for this lies in microservices' ability to more closely align with how business leaders prefer to organize and manage their teams and development processes. In other words, microservices serve as an architectural framework that better supports the desired operating model. According to a 2021 IBM survey of more than 1,200 developers and IT managers, 87 percent of microservices users agree that adopting this approach is worth the investment.

An increasingly popular organizational model involves bringing together cross-functional teams to tackle a specific business, service or product. The microservices approach fits well with this trend, allowing organizations to form small, cross-functional teams centered around a single service or group of services while promoting agile operations. The interconnected nature of microservices also includes fault isolation and application resiliency. In addition, the compact size of the services, along with their clearly defined boundaries and communication patterns, simplifies the process for new team members to understand the codebase and contribute quickly.

Microservices are common and cloud providers can benefit the most. For MSA, a cloud-based platform eliminates the need for physical infrastructure and enables the adoption of a software-as-a-service model instead of developing and maintaining software and systems in-house. The cloud is not only about infrastructure, but also about providing additional services. This enables secure MSA deployment and collaboration with ecosystem partners without increasing compliance requirements.

REFERENCES:

1. M. Fazio, A. Celesti, R. Ranjan, C. Liu, L. Chen, M. Villari, In the Cloud open problems in planning microservices. IEEE Cloud Computing. 3(5):81-88, 2016.

2. E. B. H. Yahia, L. Reveillere, Y. D. Bromberg, R. Chevalier, A. Cadot, Medley: A lightweight event-driven platform for service content. Web engineering International Conference on pp. 3-20, 2016.

3. P. D. Francesco, P. Lago, and I. Malavolta, Architecture with Microservices: a systematic mapping study. The Journal of Systems and Software 150:77-97, 2019.

4. A. Sharma, M. Kumarb and S. A. Agarwal. Software architecture A complete survey of styles and patterns. Green computing and communication 4th international conference on systems 16 - 28, 2015.

5. A. Hassan and M. Oussalah, Evolutionary Methods: Software A multi-view / multi-level model for architecture evolution. Journal of Software, 13(3):146-154, 2018.