Microservice Architecture

Abstract:

The Microservice Architecture is an architectural an organizational approach for developing software, where an application is designed as a suite of small services that communicate with each other through lightweight mechanisms such as HTTP resource APIs. Microservices are organized around business capabilities and each service is developed as independently upgradable and deployable unit. This architecture allows for decentralized management of services and teams allowing the employment of different technology stacks that are best suited for each service. The benefits provided by this architecture allows a team to overcome many problems found in monolithic applications. However, this may not suit all kinds of applications since there are also many challenges and complexities associated with it as well.

Microservice Architecture:

The microservice architecture emerged as a way of overcoming some of the major challenges faced by various software development processes. Prior to the emergence of microservices, applications were mainly developed with a Monolithic Architecture. In a monolithic application, although an application could be comprised of multiple tiers with each tier containing many components, all the components were tightly coupled and behaved as one large system. (Wu, 2017)

Building software in this manner provided multiple challenges, especially since changes were expensive with respect to cost, time and quality. Addition of features implicated that the size and the complexity of the entire application increased. Scaling an application to support increased number of users even for a single component of the application needed the entire system to be replicated. Furthermore, lots of overhead was also added to the delivery pipeline as there was a need to rebuild the whole application, rerun all test cases and deploy the whole system whenever a change occurred. (Wu, 2017) Additionally, as the product matures, the team size also increased with different teams working on tightly coupled components. This meant that the success of the application as a whole depended on the cooperation among team members, which made the team management process a challenging one as well. All these challenges resulted in slowing down the entire software development process.

The Microservice Architecture provided solutions to some of these problems. It involves separating the application into small independent functional units where each unit served a single purpose. These units communicated with each other through well-defined Application Programming Interfaces (APIs). (Bucchiarone et al., 2017)

This method of decoupling of the services implied that each functional unit could evolve independently of each other and could employ a technology stack that best suited its specific requirements. Due to the small size and complexity of each unit, modifications to the code took less time and componentization enabled each unit to be independently upgraded & replaced. The use of APIs enabled the merging of different functional units developed by multiple teams a seamless process with the deployment pipeline being managed efficiently that incorporates higher degrees of automation in deployment, testing etc. allowing the application to be deployed frequently. (Amazon, 2018)

As opposed to a monolithic application where the entire system relied on a single large database, sometimes even with multiple monoliths connecting to one DB, the microservices architecture allows each service to have its own database where the data is not directly shared unless through services that wraps the data. This gives the flexibility of adopting different DBMSs, languages etc.

based on the specific needs of each service thereby decentralizing the decision-making process for the application as well. (Bucchiarone et al., 2017)

Adoption of the microservices architecture also implies that the project teams are organized around business capabilities such as the 'Orders team', 'Shipping team' rather than their technological expertise such as the 'UI team', 'DBAs' etc. These teams held full responsibility of the respective services throughout the entire lifecycle of the service.

Many organizations have adopted the microservice architecture when building their products. Corporate giants catering to billions of users such as Amazon, Netflix, Walmart, Spotify are among the organizations that have been very successful in reaping the benefits offered by this new architecture. Some websites such as the Guardian website has incorporated this architecture into their core monolithic architecture to quickly develop and deploy new features or pages that can be added and removed whenever necessary, such as during sporting events.

Despite all these benefits it provides, microservices may not suit all applications. Decentralization and distribution of services requires time and effort invested in the initial planning process and incorporates complexity in management. When a system being built is a simple and consistent application that is not likely to undergo complex changes and caters to a stable number of users, adopting a microservice architecture is less advantageous and a monolithic implementation could be more effective and efficient. (Amazon, 2018)

One of the biggest challenges in adopting this architecture is achieving clear componentization. The success of the application depends on how well the components fits together but the separation of these services is not always an easy process as the boundaries of each service is not always very clear. Failing to achieve clear componentization increases the complexity of the boundaries because of the increased levels of communication and sharing among components. Any changes to APIs would require additional effort as the changes would impact multiple teams and the complexity of testing the application will increase as well with the need for performing tests for backward compatibility. The complexities presented by decentralized systems, versioning (Amazon, 2018) and the presence of multiple databases and the complexity of data management with respect to sharing and avoiding inconsistencies are also other big challenges that needs careful attention.

The research undertaken so far into microservices indicated that it's an emerging trend that promises to provide many benefits to an organization and its products. As with any new technology, it requires careful consideration before adopting, as it has many challenges and complexities and also might not suit all contexts. However, considering the many benefits that can be achieved through the use of the microservice architecture, this emerging trend is well worth investigating and adopting whenever appropriate.

References:

Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S. & Mazzara, M. (2017). *From Monolithic to Microservices: An experience report.* 10.13140/RG.2.2.34717.00482.

Wu, A. (2017). *Taking the Cloud-Native Approach with Microservices.* Retrieved from https://cloud.google.com/files/Cloud-native-approach-with-microservices.pdf

Amazon Web Services, Inc. (2018). *Microservices on AWS*. Retrieved from https://docs.aws.amazon.com/aws-technical-content/latest/microservices-on-aws/microservices-on-aws.pdf?icmpid=link_from_whitepapers_page