Collaborative Discussion 2 – Michael Geiger – Initial Post

A common approach for teams looking to adopt microservices is to identify existing functionality in the monolithic system that is both uncritical and fairly loosely coupled to the rest of the application (Auer et al., 2021). For example, in an e-commerce system, products and offers are often ideal candidates for a microservices proof of concept. Alternatively, more demanding teams can make the agreement that all new functionality must be developed as a microservice, which may cause a longer development time in the beginning of the transformation, but could result in a faster transformation process at the large scale.

In each of these scenarios, the key challenge is to design and develop the integration between the existing system and the new microservices. When a part of the system is redesigned with microservices, a common practice is to use glue code to interface with the new services (Jin et al., 2021). However, Washizaki et al. (2020) states that “Glue code is costly in the long term because it tends to freeze a system to the peculiarities of a specific package”. In order to solve this problem, the entire system must be restructured to a microservice-based architecture in the long term. An API gateway can help combine many individual service calls into one bigger service, thereby reducing the cost of integrating with the monolithic system (Gadge & Kotwani, 2018).

A challenge for teams starting to work with microservices are distributed transactions that span multiple independent services. In a monolithic system, this is easier because state changes are typically based on a common data model shared by all parts of the application, which is not the case with microservices (Dagger et al., 2007).

To support the transition, the main idea is to slowly replace the functionality in the system with discrete microservices while minimizing the changes that need to be added to the system itself. This is important to reduce the cost of maintaining the system and reducing the impact of the migration.

References:

Auer, F., Lenarduzzi, V., Felderer, M., & Taibi, D. (2021) From monolithic systems to Microservices: An assessment framework. *Information and Software Technology*. 137: 106600. Available from: https://www.sciencedirect.com/science/article/pii/S0950584921000793 [Accessed 25 May 2022].

Jin, Z., Zhu, Y., Zhu, J., Yu, D., Li, C., Chen, R., ... & Xu, Y. (2021) Lessons learned from migrating complex stateful applications onto serverless platforms. *Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems* 89-96. Available from: https://dl.acm.org/doi/abs/10.1145/3476886.3477510 [Accessed 25 May 2022].

Gadge, S., & Kotwani, V. (2018) Microservice architecture: API gateway considerations. Available from: http://mainlab.cs.ccu.edu.tw/presentation/pdf/(2017)Microservice-Architecture-API-Gateway-Considerations.pdf [Accessed 24 May 2022].

Washizaki, H., Uchida, H., Khomh, F., & Guéhéneuc, Y. G. (2020) Machine learning architecture and design patterns. *IEEE Softw*. 8: 1-8. Available from: http://www.washi.cs.waseda.ac.jp/wp-content/uploads/2019/12/IEEE_Software_19__ML_Patterns.pdf [Accessed 25 May 2022].

Dagger, D., O'Connor, A., Lawless, S., Walsh, E., & Wade, V. P. (2007) Service-oriented e-learning platforms: From monolithic systems to flexible services. *IEEE internet computing*. 11(3): 28-35. Available from: https://ieeexplore.ieee.org/abstract/document/4196172 [Accessed 24 May 2022].