Microservices are not a silver bullet
Benefits and Challenges of the Microservices Architectural Style

Dr. André Fachat[1]

IBM Deutschland GmbH, Mannheim, Germany
andre.fachat@de.ibm.com

Abstract

Microservices are an approach for developing and deploying services that make up an application. Microservices, as the name implies, suggest breaking down an application into smaller micro parts, deploying and running them separately. The main benefit promised by this approach is faster time-to-market for new functionalities due to better technology selection and smaller team sizes.

Experience shows, however, that Microservices are not the silver bullet in application development and projects can still fail or take much longer than expected.

To evaluate the Microservices approach, we first look at the perceived promises of the Microservices architectural style. Then we point out some of the pitfalls and challenges associated with using Microservices. In the second part we will give our opinion and experience how to set up Microservices projects to avoid these pitfalls. We will revisit and discuss the (perceived) promises of Microservices in the light of what we learned.

Microservices are a new architectural style. In 2014 Martin Fowler defined the Microservices architectural style [1], as

> an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

The Microservices architectural style is often combined with an agile and DevOps approach: small teams develop the Microservices, small enough to keep organizational efforts small. Technology selection would be up to the team, so the best technology for each job could be used. The team would also run host and support their Microservices.

What is the reasoning behind using Microservices? Traditional projects fail, but there are also many examples of successful traditional projects. So what is the business case behind Microservices?

The original value proposition promised by using Microservices can be seen in Werner Vogels presentation on the evolution of the Amazon architecture [2], and consists of time-to-market, i.e. to quickly get new features live, and to scale with increasing load requirements. We will look at these perceived benefits of Microservices: agility and faster time-to-market, more innovation, better resilience, better scalability, better reusability, and improved Return on Investment (ROI). To see why Microservices projects fail, we will briefly look at the challenges associated with the Microservices architectural style in the areas of:

- Application Architecture - trying to move traditional architectures into Microservices

- Technological Diversity - building an application using a zoo of technologies
- Team Communications - many small teams need to coordinate themselves - or be coordinated
- System Communications - networking and computing infrastructure has become even more important
- Testing and Analysis - it is not enough if every Microservice is tested in isolation
- Complexity of Operations - there are many more moving parts
- Migration - decomposing an existing application in the right order is important

We will then briefly present our best practices to avoid the pitfalls of Microservices architectures.

- New style applications - applications built for resilience, eventual consistency, graceful degradation and scalability
- Reference Tool Set and Architecture - "best technology for the job" includes long-term costs
- Feature Teams - team communication structure
- Testing and Analysis support - consolidated logging and tracing
- DevOps and Agile - feedback from users is important

We then finally revisit the Microservices promises to see which promises can be achieved, and which trade-offs have to be made. We conclude that the Microservices architectural style is a way to quickly realize small new functionalities in an agile way. If Microservices are combined to implement a larger and more complex functionality, and multiple Microservices need to cooperate, additional factors and costs need to be considered.

The work presented here is submitted to the IBM Developer web site for publication.

References

[1] J. Lewis M. Fowler. Microservices. url=https://martinfowler.com/articles/microservices.html, 2014.

[2] W. Vogels. Amazon and the lean cloud. url=https://vimeo.com/29719577, 2011.