Practices and Patterns of migration from monoliths to microservices

Luca Acquaviva[1] and Filippo Bosi[1]
[1] Imola Informatica
lacquaviva@imolainformatica.it, fbosi@imolainformatica.it

Abstract

The adoption of microservices architecture in developing business applications can bring advantages. They, however, come together with a set of challenges that have to be faced. This talk summarizes our good and bad experiences in bringing some large codebases, and all the related organizations to a more microservices-friendly approach.

1 Introduction

Often, organizations adopt microservices, only taking into account the technological and architectural side of the problem, paying little attention to the impacts that the adoption of a microservices architecture model entails.

You should anticipate the shift from a set of monoliths towards a microservices world by adopting a series of fundamental prerequisites, implementing them first of all in the monolithic architectures. And then, as a subsequent evolution, the organization will be able to extract the services from the monoliths and, working by reduction in an evolutionary fashion, will be able to transition towards a microservices architecture incrementally, without the risks usually associated to big-bang style evolutions.

In this talk we will present a collection of Patterns and Practices derived from our experiences and challenges while migrating monolithic online applications towards a “mini-services” architectural style (i.e., loosely coupled set of applications derived from the monolith, still sharing a common database). We refactored a J2EE web application towards a set of services orchestrated by Kubernetes.

We will show how it is possible to adopt an incremental path that fosters a good coexistence in production – at the same time – of the monolith alongside the sets of the soon-to-be microservices, keeping the level of the technological risk of the migration always below an acceptable level.

The strategies we adopted for managing the coexistence while allowing continuous functional changes to the production codebase range from the automation of the refactorings, to the constant automated regression testing of the refactored codebase. And resilience to functional changes while applying the migration to the management of dependencies with legacy application code that should be able to evolve during the coexistence for the entire migration timeframe.

We will show you how the biggest challenges do not arise from technological issues. They usually show up if you do not support the migration by defining correct evolutionary processes and the associated rules of conduct you will have to follow for the entire migration.

However, this is not enough. The entire organization should accept to change its processes to keep operating the new codebase effectively. And usually, the impact of the complexity of microservices in observability and management of integrations is vastly underestimated. All the actors involved in such an evolution should be aware of that, if they want their transformation to succeed.