Microservices

Introduction

Microservices is not a new term. It coined in 2005 by Dr Peter Rodgers then called micro web services based on SOAP. It became more popular since 2010. Micoservices allows us to break our large system into number of independent collaborating processes

What is a Monolith Application?

Have you ever worked in a project

- Which is released (taken to production) once every few months
- Which has a wide range of features and functionality
- Which has a team of more than 50 working for it
- Where debugging problems is a big challenge
- Where bringing in new technology and new process is almost impossible

These are typical characteristics of a Monolith applications.

Monolith applications are typically huge - more 100,000 line of code. In some instances even more than million lines of code.

Monoliths are characterized by

- *Large Application Size*
- *Long Release Cycles*
- *Large Teams*

Typical Challenges include

- *Scalability Challenges*
- *New Technology Adoption*
- *New Processes - Agile?*
- *Difficult to Automation Test*
- *Difficult ot Adapt to Modern Development Practices*

Microservices

Microservice Architectures evolved as a solution to the scalability and innovotation challenges with Monolith architectures.

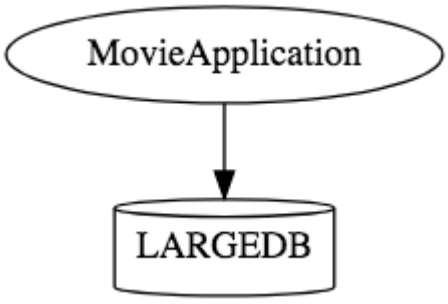
There are a number of definitions proposed for Microservices

While there is no single accepted definition for microservices, for me, there are a few important characteristics:

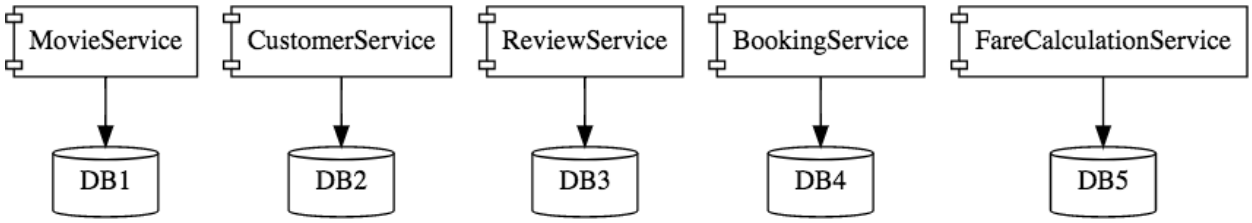
- *REST - Built around RESTful Resources. Communication can be HTTP or event based.*
- *Small Well Chosen Deployable Units - Bounded Contexts*
- *Cloud Enabled - Dynamic Scaling*

How does Microservice Architecture look like?

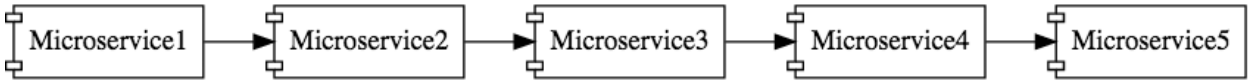
This is how a monolith would look like. One application for everything.



This is how the same application would look like when developed using Microservices Architecture.



Microservice Architectures involve a number of small, well designed, components interacting with messages.



Advantages of Microservices

Advantages

- *New Technology & Process Adaption becomes easier. You can try new technologies with the newer microservices that we create.*
- *Faster Release Cycles*
- *Scaling with Cloud*

Challenges with Microservice Architectures

While developing a number of smaller components might look easy, there are a number of inherent complexities that are associated with microservices architectures.

Lets look at some of the challenges:

- *Quick Setup needed : You cannot spend a month setting up each microservice. You should be able to create microservices quickly.*
- *Automation : Because there are a number of smaller components instead of a monolith, you need to automate everything - Builds, Deployment, Monitoring etc.*
- *Visibility : You now have a number of smaller components to deploy and maintain. Maybe 100 or maybe 1000 components. You should be able to monitor and identify problems automatically. You need great visibility around all the components.*
- *Bounded Context : Deciding the boundaries of a microservice is not an easy task. Bounded Contexts from Domain Driven Design is a good starting point. Your understanding of the domain evolves over a period of time. You need to ensure that the microservice boundaries evolve.*
- *Configuration Management : You need to maintain configurations for hundreds of components across environments. You would need a Configuration Management solution*
- *Dynamic Scale Up and Scale Down : The advantages of microservices will only be realized if your applications can scaled up and down easily in the cloud.*
- *Pack of Cards : If a microservice at the bottom of the call chain fails, it can have knock on effects on all other microservices. Microservices should be fault tolerant by Design.*
- *Debugging : When there is a problem that needs investigation, you might need to look into multiple services across different components. Centralized Logging and Dashboards are essential to make it easy to debug problems.*

- *Consistency : You cannot have a wide range of tools solving the same problem. While it is important to foster innovation, it is also important to have some decentralized governance around the languages, platforms, technology and tools used for implementing/deploying/monitoring microservices.*

Solutions to Challenges with Microservice Architectures

Spring Boot

Enable building production ready applications quickly

Provide non-functional features

- *embedded servers (easy deployment with containers)*
- *metrics (monitoring)*
- *health checks (monitoring)*
- *externalized configuration*

Spring Cloud

Spring Cloud provides solutions to cloud enable your microservices. It leverages and builds on top of some of the Cloud solutions opensourced by Netflix (Netflix OSS).

Important Spring Cloud Modules

Dynamic Scale Up and Down. Using a combination of

- *Naming Server (Eureka)*
- *Ribbon (Client Side Load Balancing)*
- *Feign (Easier REST Clients)*

Visibility and Monitoring with

- *Zipkin Distributed Tracing*
- *Netflix API Gateway*

Configuration Management with

- *Spring Cloud Config Server*

Fault Tolerance with

- *Hystrix*