Does adopting microservices help with continuous delivery?

Microservice is a software development architecture which is componentized by breaking down to many small and independent services, which are built around business functions and can be deployed independently through an automated deployment mechanism[1].Using microservice can help with continuous delivery.

Compared to monolithic architecture, microservice architecture is more scalable in terms of business and technology. Microservice divides a complex application into separate services, and the services interact in a loosely coupled form. In continuous delivery, microservice provides many benefits over monolithic architectures. Microservice can remove single points of failure by ensuring that errors in one service do not crash or impact other parts of an application[2]. Individual microservice can be scaled out independently to provide additional availability and capacity. One of the representative examples is Netflix company[3], who adopts microservice to improve deliver more features to their customers, and much faster. Netflix developers build applications as suites of services that can be deployed separately, and allows different services to be written in different programming languages[3].The benefit is that developers can update an existing service without rebuilding and redeploying the entire application and quickly deliver new features to their customers.

Using microservices can increase team efficiency in the continuous delivery. In a monolithic architecture, some changes made to the application may take more time because of the impact on the entire system[1]. However, if developers split a monolithic system into microservices, each change introduced to the system can only affect an individual service[4]. Therefore, developers only need to deploy the changed service. For example, implementing an update on a monolithic system will require a 10-minute unit test, 2-hour acceptance test and 20-minute deployment. Because a small change requires the entire monolith to be rebuilt and deployed. By contrast, the same updates in a microservice system only need a 1-minute unit test, 1-minute integration test and 5-minutes deployment[4].

Although microservice provide significant supports for independent development and deployment, it has posed enormous challenges to the organization and technical layers in implementing continuous delivery[4]. Since services are separate, the independent deployment of each service requires a lot of effort and cost. In microservice architecture, different services need to adopt different technologies based on corresponding demands, which also introduce huge challenges.

References

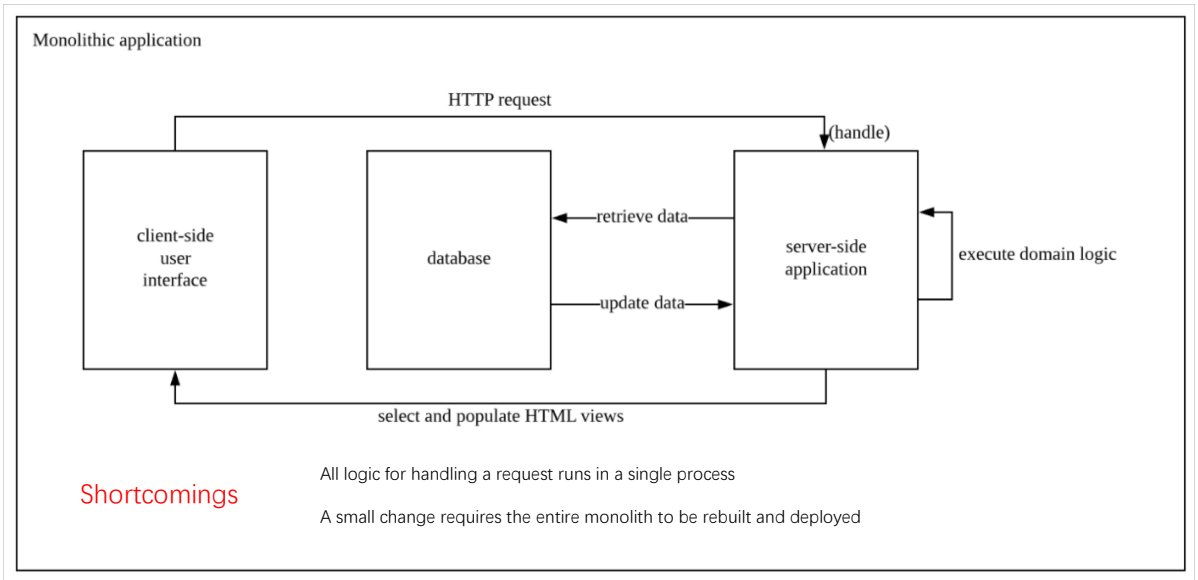
[1] Fowler M., Lewis J. **Microservices** https://martinfowler.com/articles/microservices.html *Date Accessed: 8 February 2019*

[2] Guckenheimer S. **What are Microservice?** https://docs.microsoft.com/en-us/azure/devops/learn/what-are-microservices *Date Accessed: 8 February 2019*

[3] Weaveworks **The Shift to Microservices and Continous Delivery** https://www.weave.works/blog/microservices-and-continuous-delivery *Date Accessed: 8 February 2019*

[4] Alibaba Cloud **Continuous deployment with microservices** https://medium.com/@Alibaba_Cloud/continuous-deployment-with-microservices-f259dcc60618 *Date Accessed: 8 February 2019*

Monolithic



Shortcomings

All logic for handling a request runs in a single process

A small change requires the entire monolith to be rebuilt and deployed

Microservice

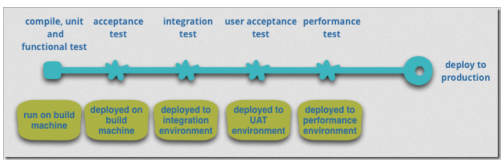
Smaller granularity of services can make it easier to create relationships between developers and users

Martin Fowler

1. Microservice componentizes software by breaking down into **service**
2. Microservice is choreographed using simple RESTish protocols (HTTP request-response with API, lightweight messaging)
3. Decoupled and cohesive: applications own their own domain logic and can be independently deployable
4. Decentralized governance
5. Services can be implemented in various languages, and data management is decentralized

Ethos

Products not projects	Microservice teams would expect to see sophisticated monitoring and logging setups for each individual service such as dashboards showing up/down status and a variety of operational and business relevant metrics.
You build, you run it	



Continuous integration & Continuous delivery pipeline

Risks

Not enough time has passed to judge microservices are the future direction for software architectures

It's hard to figure out exactly where the component boundaries should lie

If the components do not compose cleanly, then all you are doing is shifting complexity from inside a component to the connections between components.