MICROSERVICES IN HEALTHCARE

Granulate to Accelerate!

VICERT

Content

What are Microservices and why should you consider implementing them

Microservices is almost a philosophy, as it is used to develop applications not piece by piece but by building all the pieces simultaneously by different teams. It is a special implementation approach for service-oriented architectures (SOA) used to build flexible, independently deployable software systems.

Shortening the time to market, but also allowing for the system to be constantly improved and in theory much more scalable. Services in the microservice architecture are processes that mutually communicate over the network in order to accomplish their own distinctive goals. They need to be fine grained (stripped down to basic business functions) while the protocols have to be lightweight (for example: uploading photos or creating a user profile in a library).

The microservice architecture enables the continuous deployment/delivery of complex applications, thus enabling organizations to improve their tech stack (on the go).

Philosophy and principles

The philosophy of microservices equals the Unix philosophy and is explained with one simple sentence:

Do one thing and do it well.

It is described as follows:

- Each microservice is elastic, volatile, composable, minimal, and complete.
- The services are small & fine-grained to perform a single function.
- The organization should embrace automation of testing and deployment, thus allowing for different development teams to work on independently deployable units of code.

Why use microservices and not monolith

A monolithic application describes a single-tiered software application. Its characteristic is the user interface and data access code are combined into a single program from a single platform. Although it was popular before, now there are a number of reasons why you should move to microservices.

The downsides of using Monolith Architecture are:

- It is limited in size and complexity.
- Monolith Applications are too large and complex to make changes fast with quality.
- The size of the application can slow down the start-up time.
- Developers have to redeploy the entire application on each update.
- Continuous deployment is difficult.
- Monolithic applications have difficulties adopting new tech. Changes in frameworks or languages will affect the entire application, thus making it very expensive.

All of the problems that monolithic applications face can be resolved with a transfer to microservices.

Microservices benefits - Top 7 reasons

1. They resolve the problem of complexity by decomposing an application into a set of smaller and more manageable services that are faster to develop and easier to understand and maintain.

2. They allow for each service to be developed independently by a team that is solely focused on that service.

3. They remove and reduce barriers of adopting new technologies with polyglot programming. This allows developers the freedom to choose technologies that they will use for the service they are making.

4. They enable each microservice the ability to be deployed independently. As a result, it makes continuous deployment possible for complex applications and for each service to be scaled independently.

5. They are easy to integrate with third-party solutions.

6. They simplify security monitoring because many different parts of an application are isolated. A security issue could happen in one part without affecting other parts of the project.

7. They have good fail-safe protection, if one microservice stops working, the other ones will continue to work.

How to start using microservices

1. Identify potential microservices categories where you may find value.
2. Define the breadth of responsibility for the identified microservices.
3. Consider the type of information that will be transmitted.
4. Associate business processes with the technical functionality defined.
5. Link tech processes to the business processes.
6. Research capabilities that are not offered today but are desired.
7. Design the microservice starting with the API definition and elaborate how the service will be consumed.
8. Develop a service mocking or simulation.
9. Deploy the microservice.
10. Manage container systems.

Microservices in Healthcare

In order to accelerate healthcare modernization, microservices seem like the right way to go. Modernizing app development and speeding up the go to market time does seem like an appealing way to go.

For example: A patient portal can be managed with multiple services and then scaled independently across different instances, all sharing one central database. It would drastically cut the cost of building a portal, maintenance, as well as enable further scalability.

However, we have to keep in mind the security aspect.
If for example we were to tackle the issues we have now with EHRs and the snowball effect that turned them into insanely large and complex systems, implementing Microservices architecture would definitely help with manageability of certain applications, speed up their runtime, as well as enable better movement between infrastructures. Additionally, this would enrich the data analysis, as it would enable data to be processed globally, simultaneously correlating live with legacy and public data.

The Inevitable Apache Kafka

Do not use microservices without Kafka. The tendency of microservice building blocks is lack of communication. It is very difficult to track if there was a problem, or a protocol was not completed. Since they are so granulated they do not carry the function of reporting if a problem occurs, so sometimes investigating where the problem happened can take more time than fixing it. The solution is to use Kafka that allows you to easily disassociate communication between various microservices.

Kafka stores, receives and sends messages on different brokers. The main benefits from this approach are great scalability and fault tolerance.

Along with Streams it makes it fairly easy to write business logic that enriches Kafka data for service consumption. Its possibilities of handling enormous amount of events per day, low latency and error tolerance makes it extremely powerful and popular among corporations worldwide.

Tips and tricks

- Compare polyglot & monoglot: build a microservice using the language that suits it best for the requirements its solving, while keeping it best suited for the team that is working on it.
- Give power to developers: Make sure that your developers have everything they need in order to build the systems and keep them involved in all parts of the delivery and deployment process.
- Granulation first: The more granular a microservice is, the easier is to change it or drop it.
- Make it scalable: Functionality is important, but what happens if you receive more requests than you can handle?!
- Security is NOT overrated: Patch your system properly! Always keep in mind that vulnerabilities can pop out!
- Automation is your friend: For your teams moving fast, you need to automate release process!

What our developers had to say about the future of Microservices

Further development of microservices will replace developers in the future - one of our senior developers who did not want to be named, but is definitely dreaming of an early retirement while his microservices bring home the paychecks ;)

Next year will be the year of microservices - Our Senior Team Leader who embraced this project as if it was a chocolate covered muffin with ruby <3

After hearing what the first developer said, our Microservices expert (who is also the founder of Microservices weekly) stated that: "Microservices will have so many problems that the role of developers will be ever so important".

What is your opinion? Tweet us :)

Additional resources (references)

1. http://microservices.io/index.html
2. https://www.cio.com/article/3159071/innovation/microservice-ecosystems-for-healthcare.html
3. https://martinfowler.com/articles/microservices.html
4. https://www.researchgate.net/profile/Kevin_Khanda/publication/317820124_Microservices_Science_and_Engineering/links/5967819fa6fdcc18ea66241a/Microservices-Science-and-Engineering.pdf
5. https://medium.com/tag/microservices
6. https://kafka.apache.org/
7. https://dzone.com/articles/micro-services-for-performance

ENABLING THE DIGITAL HEALTH REVOLUTION!

CONTACT

www.vicert.com
info@vicert.com
@vicert_inc

SAN FRANCISCO

1355 Market Street, Suite 488
San Francisco, CA 94103, USA
+1.415.495.7700

VICERT