Lab Assignment 1 CMT

Max Shahrokni

TASK 5

Microservices

- Benefits and Disadvantages compared to a Monolithic architecture

Lets start off with the benefits:

• The microservice architecture is modular in approach, i.e consists of smaller services that together make a whole. Therefore each service may have a team assigned to it, which therefore makes on-boarding new developers a shorter, more straightforward process compared to a monolithic structure.

• Microservices architecture allows each feature to be developed and deployed independently, meaning that changes can be made on one service without risking the entire application failing. Better fault tolerance and isolation.

• Monolithic architecture restrict data storage options to just one location which isnt very scalable, while microservices allows for the choice of the storage option that is best suited.

• Another benefit for the microservices architecture over the monolithic one is that it allows for polyglotic programming. The individual components may be programmed in different languages, that is best suited for the task.

• Deployment speed is increased since the microservice architecture aligns well with with CI/CD, DevOps philosophy.

Moving on to the disadvantages:

- Microservices involve a lot more moving parts than traditional applications that requires enormous effort and careful planning to run smoothly.

- Microservices are often more expensive than monoliths. Since services will need to communicate with each other, leading to higher volume of remote calls, which may increase network latency and processing costs.

- Also in comparison to monolithic architecture, microservices comes with some security challenges due to the sharp increase in the volume of data exchanged between modules.

- Requires a culture change for companies looking to switch over from a monolithic structure, which is expensive and time consuming.

- Suitable use case scenario

Big data applications that collect, ingest, process and deliver big data in a pipeline oriented structure. Microservices naturally fit within this type of architecture since each step in a data pipeline handles one small task.

- Two or more architectural patterns explained

Bulkhead Pattern

Isolate elements of an application into pools so that if one fails, the others will continue to function. This pattern is named Bulkhead because it resembles the sectioned partitions of a ship’s hull. Partition service instances into different groups, based on consumer load and availability requirements. This design helps to isolate failures, and allows you to sustain service functionality for some consumers, even during a failure.

Sidecar Pattern

Deploy components of an application into a separate processor container to provide isolation and encapsulation. This pattern can also enable applications to be composed of heterogeneous components and technologies. This pattern is named Sidecar because it resembles a sidecar attached to a motorcycle. In the pattern, the sidecar is attached to a parent application and provides supporting features for the application. The sidecar also shares the same lifecycle as the parent application, is created and retired alongside the parent. The sidecar pattern is sometimes referred to as the sidekick pattern and is the last decomposition pattern that we show in the post.