<마이크로 서비스 아키텍쳐(MSA)>

MSA 는 소프트웨어 개발 기법 중 하나로, 어플리케이션 단위를 '목적'으로 나누는 것이 핵심

- **Monolithic vs MSA**

MSA 가 도입되기 전, Monolithic 아키텍처 방식으로 개발이 이루어졌다. Monolithic 의 사전적 정의에 맞게 '한 덩어리'에 해당하는 구조로 이루어져 있다. 모든 기능을 하나의 어플리케이션에서 비즈니스 로직을 구성해 운영한다. 따라서 개발을 하거나 환경설정에 있어서 간단한 장점이 있어 작은 사이즈의 프로젝트에서는 유리하지만, 시스템이 점점 확장되거나 큰 프로젝트에서는 단점들이 존재한다.

- 빌드/테스트 시간의 증가한다.
- 작은 문제가 시스템 전체에 문제를 일으킨다.
- 확장성에 불리 하다.

MSA 는 좀 더 세분화 시킨 아키텍처라고 말할 수 있다. 한꺼번에 비즈니스 로직을 구성하던 Monolithic 방식과는 다르게 기능(목적)별로 컴포넌트를 나누고 조합할 수 있도록 구축한다.

MSA에서 각 컴포넌트는 API를 통해 다른 서비스와 통신을 하는데, 모든 서비스는 각각 독립된 서버로 운영하고 배포하기 때문에 서로 의존성이 없다. 하나의 서비스에 문제가 생겨도 다른 서비스에는 영향을 끼치지 않으며, 서비스 별로 부분적인 확장이 가능한 장점이 있다.

즉, 서비스 별로 개발팀이 꾸려지면 다른 팀과 의존없이 팀 내에서 피드백을 빠르게 할 수 있고, 비교적 유연하게 운영이 가능할 것이다.

좋은 점만 있지는 않다. MSA 는 서비스 별로 호출할 때 API 로 통신하므로 속도가 느리다. 그리고 서비스 별로 통신에 맞는 데이터로 맞추는 과정이 필요하기도 하다. Monolithic 방식은 하나의 프로세스 내에서 진행되기 때문에 속도 면에서는 MSA 보다 훨씬 빠를 것이다. 또한, MSA 는 DB 또한 개별적으로 운영되기 때문에 트랜잭션으로 묶기 힘든 점도 있다.

따라서, 서비스별로 분리를 하면서 얻을 수 있는 장점도 있지만, 그만큼 체계적으로 준비돼 있지 않으면 MSA 로 인해 오히려 프로젝트 성능이 떨어질 수도 있다. 정답이 정해져 있는 것이 아니라, 프로젝트 목적, 현재 상황에 맞는 아키텍처 방식이 무엇인지 설계할 때부터 잘 고민해서 선택해야 한다.

Reference

1. https://shaul1991.medium.com/%EC%B4%88%EB%B3%B4%EA%B0%9C%EB%B0%9C%EC%9E%90-%EC%9D%BC%EC%A7%80-%EB%8C%80%EC%84%B8-msa-%EB%84%88-%EB%AD%90%EB%8B%88-efba5cfafdeb

2. http://clipsoft.co.kr/wp/blog/%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C%EC%84%9C%EB%B9%84%EC%8A%A4-%EC%95%84%ED%82%A4%ED%85%8D%EC%B2%98msa-%EA%B0%9C%EB%85%90/