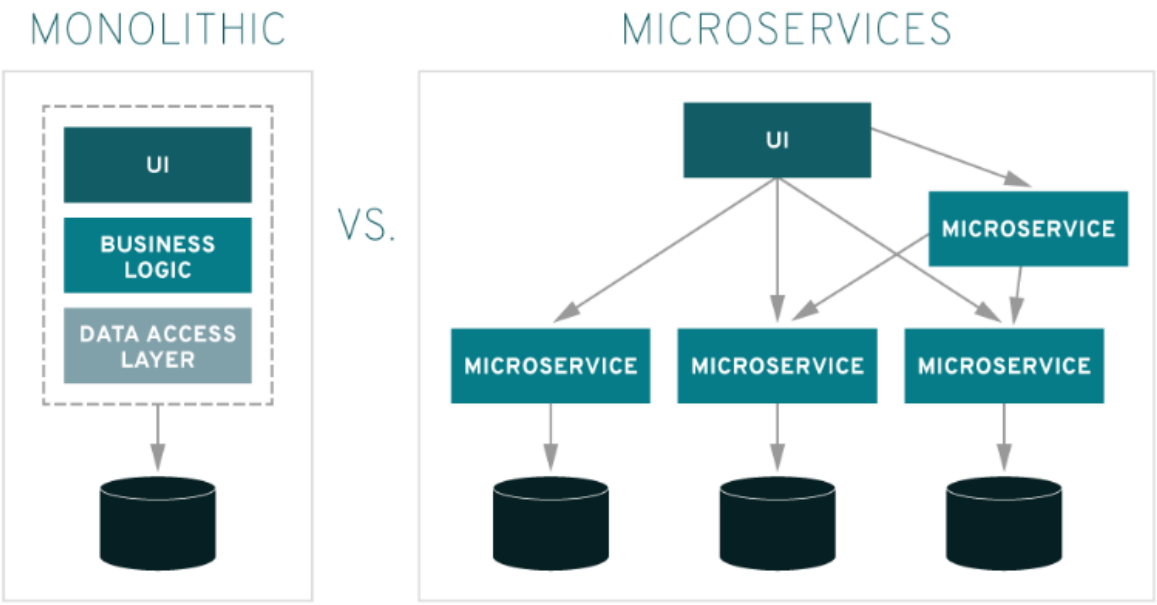
마이크로서비스란?



1. 개념

```
마이크로서비스는 애플리케이션 구축을 위한 아키텍처 기반의 접근방식이다.
전통적인 모놀리식(monolithic) 접근방식과 구별짓는 기준은 애플리케이션을 핵심 기능으로 세분화하는 방식이다.
개별 서비스가 다른 서비스에 부정적 영향을 주지 않으며 작동한다.
(웹서비스에서 REST API로 서버를 따로 구축하는 것과 비슷하다고 생각한다.)
```

2. 모놀리식 vs 서비스 지향 아키텍처(SOA)

```
모놀리식은 전체 애플리케이션의 소스는 하나의 배포유닛으로 통합된다. (war 등)
따라서 일부 오류 및 업데이트 등의 작업 시 전체 기능을 오프라인으로 전환하고 문제를 해결해야 한다.
소규모에선 가능하지만 규모가 커질수록 불편한 구조입니다.

반면 서비스 지향 아키텍처(SOA)는 애플리케이션을 별개의 재사용 가능한 서비스 단위로 분할하여 엔터프라이즈 서비스 버스(ESB)를 통해 통신한다.
통합하여 하나의 애플리케이션을 구성한다.
따라서! A,B,C의 기능 중 C를 업그레이드하고싶다면 C기능의 서버를 비활성화하고 기능을 잠시 사용불가상태로 만들면 된다.
유지보수에 탁월한 장점이 있다.
```

3. SOA와 마이크로서비스

[차이점]
마이크로서비스는 스테이트리스(stateless)방식으로 서로 통신할 수 있으므로 이러한 방식으로 구축된 애플리케이션은 내결함성이 더 높고 단일 ESB에 대한 의존성은 더 낮다.
마이크로서비스가 언어에 구속받지 않는 API이기 때문에 개발팀이 자체 툴을 선택할 수 있다!!

SOA의 본질을 생각했을 때 마이크로서비스는 완전히 새로운 개념은 아니다.

따라서 본질은 같으나 API로 구현한 마이크로서비스는 개발, 유지보수에 가용성이 높다는 것이다.

4. 마이크로서비스의 장점

마이크로서비스는 분산형 개발을 통해 업무 능력의 향상을 불러온다.
동시에 여러 기능에 대한 마이크로서비스를 개발하는 것이 가능하다.
따라서! 더 많은 개발자가 동시 참여할 수 있으므로 개발, 유지보수의 능률이 향상된다.

출시 기간 단축, 높은 확장성, 복구력, 손쉬운 배포, 편리한 액세스, 개방성 등의 장점을 가진다.

5. 구축을 위한 노력

[종속성]
서비스 간의 종속성을 파악해야한다.

[테스트]
통합테스트, 엔드투엔드 테스트를 수행해야한다.
분리해 구현한 서비스가 서로 지원하도록 구성되어야한다.
아키텍처 한 부분의 장애가 전체의 장애를 불러올 수 있다.

[버전관리]
새로운 버전으로 업데이트 할 때는 이전 버전과의 호환성문제를 고려해야한다.

[배포]
편리한 배포를 위해선 상당한 수준의 자동화에 투자해야한다.
마이크로서비스의 복잡성에 의해 수동배포가 어려울 수 있다.

[로그관리]
분산 시스템에서 모든 내용을 한 곳에 모을 수 있는 중앙집중식 로그가 필요하다.

[모니터링]
분산되어있는 서비스이기 때문에 문제의 근원을 정확히 짚어내야한다.

[디버그]
원격 디버그는 선택이 아니라 필수이다.
시스템을 중앙에서 파악할 수 있는 능력이 필요하다.

6. 내 생각

개인에게 있어 마이크로서비스를 구현하는 것은 어려움이 있을지도 모르겠다.
서비스 내 여러 기능에 대한 서버를 관리하고 이를 통합하고 배포해야만한다.
가상환경을 활용해 분리해 개발한 기능을 통합하는 연습을 해봐야겠다.

마이크로서비스로 전환되고있는 추세에 맞춰 연습해야겠다.
역시 연습은 Spring으로..

Made by 장경석