1.1 쿠버네티스와 같은 시스템이 필요한 이유

생성일	@2021년 3월 11일 오후 10:33
태그	

1. 모놀리스 앱이 마이크로서비스로 전환

모놀리스의 단점

- 조금만 변경해도 전체 앱을 재배포해야함
- 각 구성간의 상호의존성 제약 커져서 결국 전체 시스템의 복잡성 증가
- 시스템의 수직 확장시 비용문제 가 있다
- 시스템의 수평 확장시 여러가지 제약과 한계 존재

시스템 수직확장(scale up) : CPU, 메모리, 서버 구성요소를 추가
시스템 수평확장(scale out) : 서버를 추가, 앱의 복제본 실행

마이크로서비스로 애플리케이션 분할

- 마이크로서비스는 모듈단위의 독립적으로 배포할 수 있는 작은 구성 요소로 분할
- 각각 API로 다른 마이크로서비스와 통신 (Restful, AMQP, gRPC...)

마이크로서비스 확장

- 마이크로서비스의 확장 : 서비스별로 수행된다 → 리소스가 더 필요한 서비스만 별도로 수평/수직 확장

마이크로서비스 배포

마이크로서비스의 단점

- 구성요소가 많아질 시, 배포 조합의 수 + 상호 종속성 모두 고려해서 결정해야함
- 마이크로서비스 배포 구성/디버깅의 어려움 존재

환경 요구사항의 다양성

- 동일한 서버에서 실행중인 다양한 앱간의 라이브러리 버젼관리 + 종속성 충돌 가능성 고려해야함
    - 서로 다른 버전의 동일한 라이브러리를 필요로 하는 경우 종속성 충돌

2. 애플리케이션에 일관된 환경 제공

개발 ↔ 프로덕션 환경 사이의 차이를 줄여야한다 → 동일한 환경을 만드는 것이 제일 이상적

3. 지속적인 배포로 전환: 데브옵스와 노옵스

데브옵스란?
개발하는 팀이 앱을 배포(CD)하고 관리하는것
→ 개발 + QA + 운영 팀이 전체 프로세스에서 협업
→ 원래는 데이터센터와 운영담당자 / 개발자 이렇게 정보 격차가 있었음

쿠버네티스가 바로 데브옵스(+노옵스) 를 가능하게 합니다

- 쿠버네티스가 하드웨어를 추상화 하고 이를 애플리케이션 배포, 실행을 위한 플랫폼으로 제공