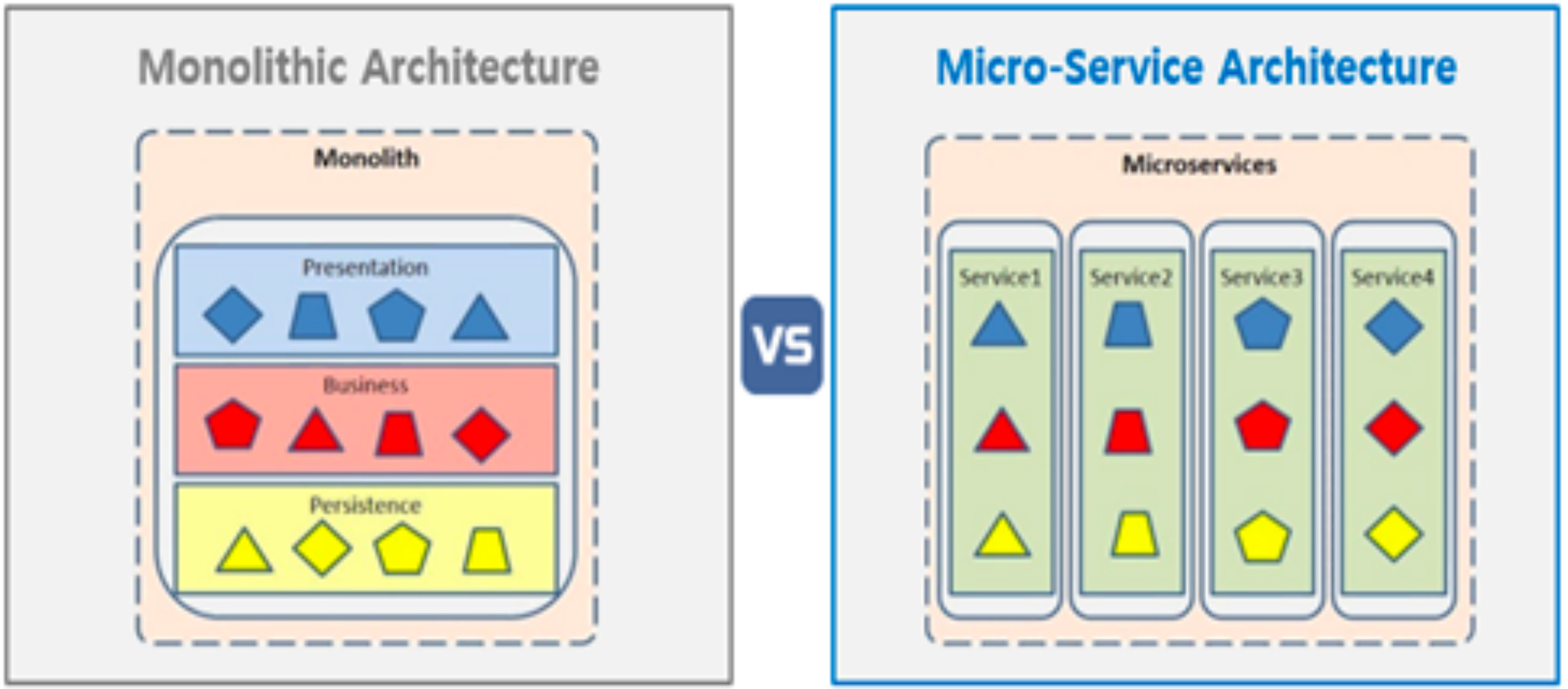
# MSA

우준성 (whitebear)

# 🤔 MSA? 먹는거야?

# Micro Service?

작고, 독립적으로 배포 가능한
각각의 기능을 수행하는
서비스로 구성된 프레임워크




※ Micro-Service : 한 개 이상의 서비스(기능)로 Interface - Business Logic - Data 영역 모두 포함하여 독립적인 개발/수정/배포가 가능한 컴포넌트

# 모놀리식 아키텍쳐

소프트웨어의 모든 구성요소가
한 프로젝트에 통합되어 있는 형태

Auth 모듈 개발
Posting 모듈 개발
....

# Auth 모듈 개발
# Posting 모듈 개발

....

하나의 결과물로 패키징

주로 소규모 프로젝트에 사용

# 왜?

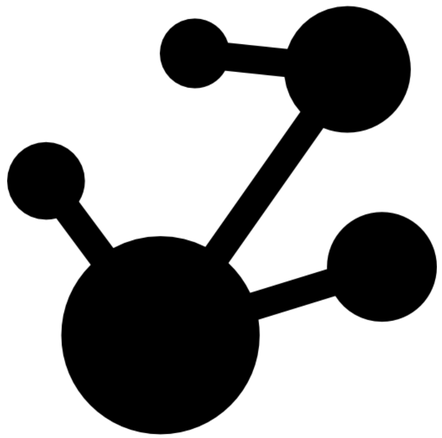
일정 규모 이상
수백명 이상의 개발자 투입

# 부분적인 장애 → 전체 서비스의 장애

# 부분적인 Scale Out이 어려움

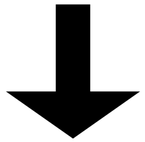
여러 서버로 나누어
일을 처리하는 방식

여러 컴포넌트 강하게 결합 → 서비스 변경 어려움



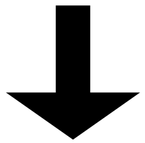
# 한 Framework와 언어에 종속적

On-premise 서버 기반 Monolithic Architecture



클라우드 환경 기반 MSA

# Monolithic Architecture

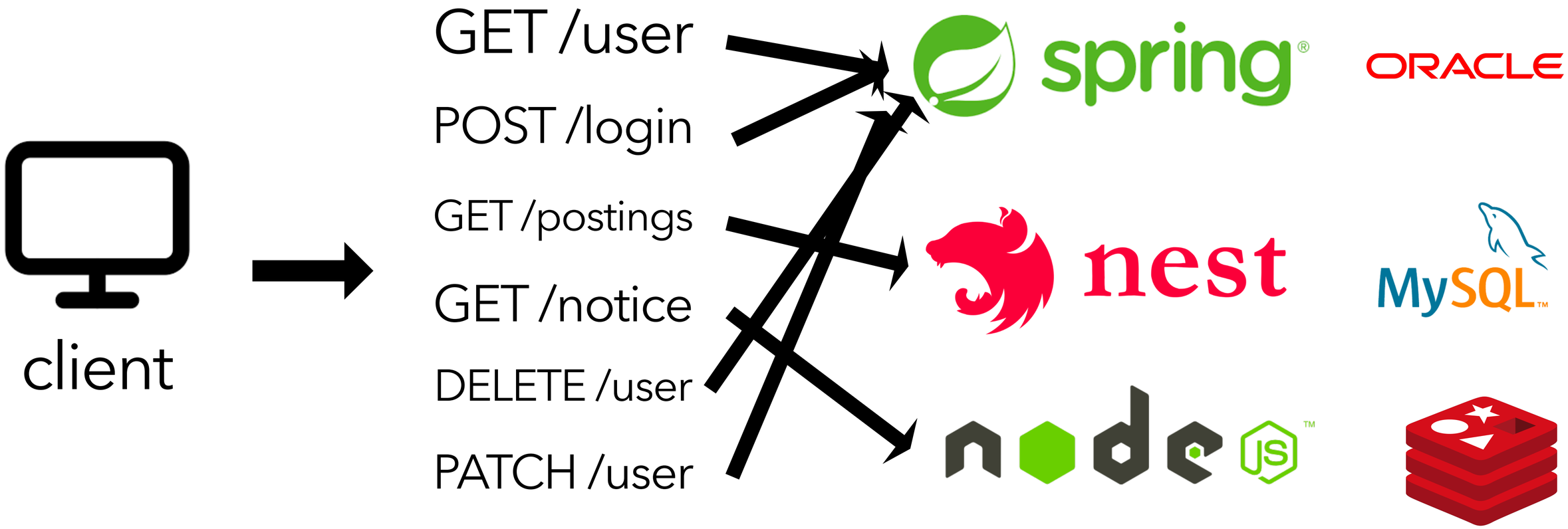


# MSA

# MSA는 API를 통해서만 상호작용

서비스의 End-point를 API 형태로 외부에 노출하고
실질적인 세부 사항은 모두 추상화

내부의 **구현 로직, 아키텍쳐와 프로그래밍 언어, 데이터베이스, 품질 유지 체계**
와 같은 기술적인 사항들은 서비스 API에 의해 철저히 가려짐


client
GET /user
POST /login
GET /postings
GET /notice
DELETE /user
PATCH /user
spring
ORACLE
nest
MySQL
node
JS

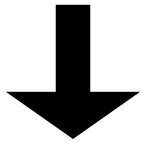
마이크로 서비스는 하나의 기능만 수행 (책임)

기술 중립적 프로토콜 사용 → 서비스 구현 기술과는 무관

무거운 제품에 의존 X  (REST, Kafka 등 사용)

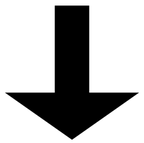
# 🧐 그래서 무슨 장점이 있는데요?

각각의 서비스는 모듈화 → 모듈끼리 PRC 또는 message-driven API 이용



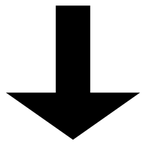
개별의 서비스 빠르게 개발, 유지보수도 쉬움

팀 단위로 적절한 기술 스택을 다르게 가져갈 수 있음



서로 프레임워크(기술)의 장점을 극대화해서 개발

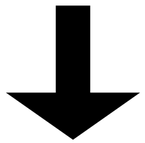
서비스별로 독립적으로 배포 가능



지속적인 배포 CD도 모놀리식에 비해 가벼움

Continuous Deployment

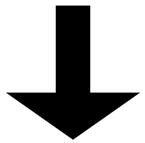
각각 서비스의 부하에 따라 개별적 Scale-out 가능



메모리, CPU 부분으로 이득

# 그럼 이제 모두 MSA 쓰면 되는건가요?

MSA는 모놀리식 구조에 비해 많이 복잡

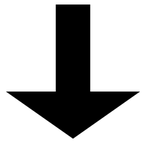


개발자는 내부 시스템에 대해 고민해야 함

통신 장애/서버 부하의 경우 어떻게 Transaction을 유지할지 결정

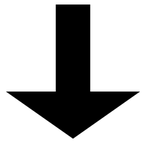
MSA에서는 비즈니스 로직에 대한 DB 각기 다름

서비스의 연결을 위해 통신이 포함되어 있음



트랜잭션 유지 어려움

통합 테스트 어려움



개발 환경과 실제 운영 환경을 동일하게 가져가기 어려움

실제 운영환경에 대해 배포하는 것이 쉽지 않음

다른 서비스와의 연계가 정상 작동하는지도 확인해야 함

참조

https://wooaoe.tistory.com/57

https://velog.io/@tedigom

https://sabarada.tistory.com/51?category=822070

# 결론!

# 각각의 상황에 맞게 아키텍쳐 알잘딱깔센

감사합니다 😆