

마이크로서비스 아키텍처 이해하기

--운영의 관점에서--

클라우드 서비스팀 남세현

여러분이 만나는 시스템은 어떤 시스템입니까?

- 운영/관리가 편한 시스템
- 스케일링이 편한 시스템
- 발 뻘고 편안히 잠을 잘 수 있는 시스템
- 새로운 실험, 테스트를 편히 할 수 있는 시스템

혹은

- 운영/관리가 힘든 시스템
- 스케일링이 힘든 시스템
- 자동화가 안되어있는 시스템
- 빌드/테스트/릴리즈 사이클이 느린 시스템

그런 여러분의 시스템은

퍼블릭 클라우드에 적합한 시스템입니까?

왜 퍼블릭 클라우드로 가야하나요?

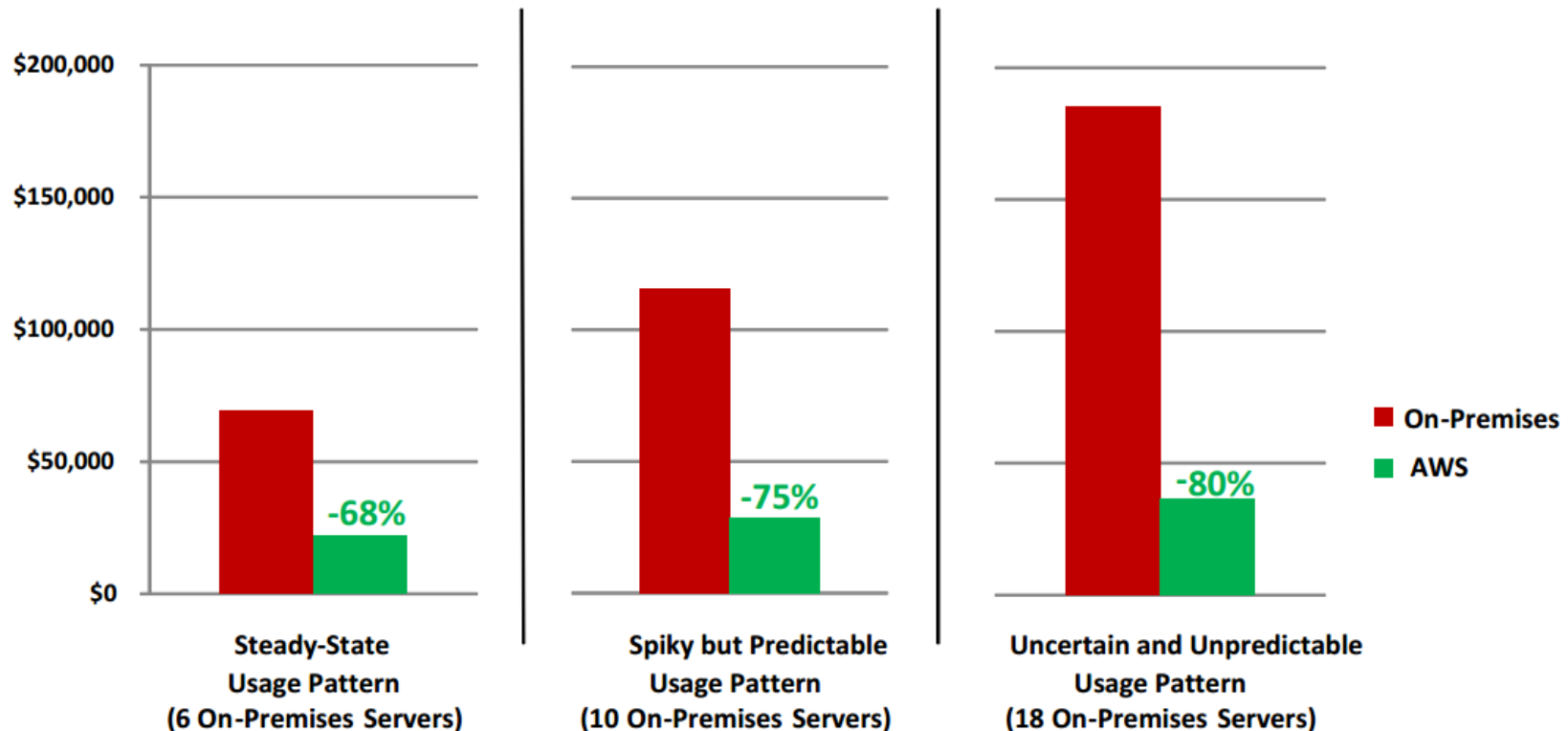
Q. 퍼블릭 클라우드를 사용하는 이유는?

퍼블릭 클라우드를 사용하는 이유

- 잘 만들어진 Managed Services
 - "바퀴를 다시 발명하지 말라"
 - Computing, Storage, 자동화, Log, etc...
- 인프라의 관리, 보안, 공급(provision) 제공
 - 장비 망가져서 교체하러 출장 갈 필요 X
 - 은행권도 사용할 정도의 보안레벨
 - 장비 공급에 한달 기다릴 필요가 없음
- 업계 최고 레벨의 기술력 제공

퍼블릭 클라우드를 사용하는 이유

- 높은 퀄리티를 적은 비용으로
 - 99.95 Availability, 99.9999999999 Durability (11 9's)
 - SLA(Service Level Agreement)
 - 규모의 경제, 유연한 인프라 증감(Pay as you go)



TCO of Web Applications (Compute and Database) for 3 Years

Q. 그렇다면 우리의 시스템도 AWS의 장점을 누릴 수 있는가?

- 운영/관리 힘든 시스템 + 운영/관리 편한 인프라 (?!)
- 스케일링 힘든 시스템 + 스케일링 편한 인프라 (?!)
- 탄력성이 적은 시스템 + 탄력적인 비용의 인프라 (?!)

오늘은 우리의 시스템이 어떻게 변해야 하는지에 대해 이야기 해볼
까 합니다.

우리의 시스템 - Monolithic 이란?

"단단히 짜여 하나로 되어 있는"

하나의 프로세스에 모든 기능이 들어있는 Architecture

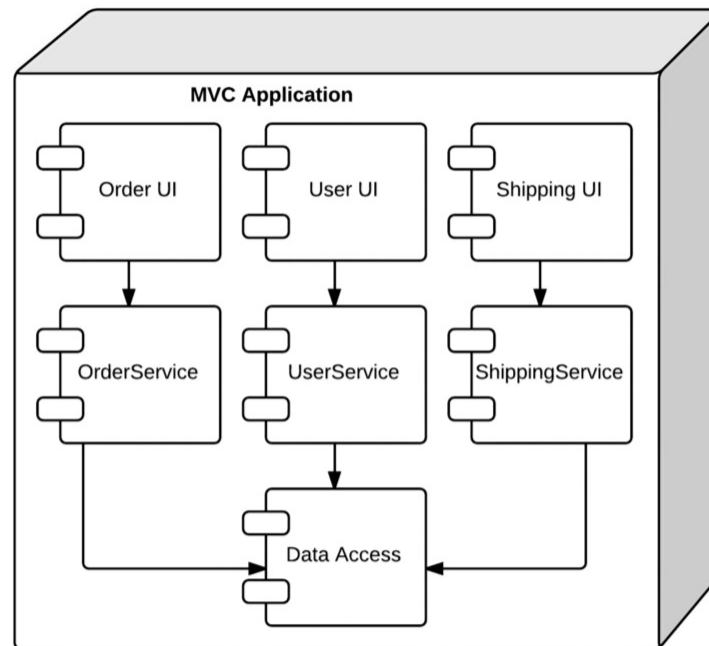


Figure 4: Monolithic Architecture

Monolithic의 문제점

- 빌드/테스트/릴리즈 사이클이 너무 느림
 - 빌드만 수 시간 이상 걸림
 - 하나의 코드베이스 = 많은 커밋 충돌
 - 한달 1~2번 밖에 릴리즈 못함
 - 실험, 혁신은 존재할 수 없는 환경
- 기능 추가가 너무 힘들
 - 강한 소프트웨어 결합성
 - 개발을 위해 알아야 할게 너무 많음
 - 어디에 어떻게 영향을 미칠지 몰라서...

Q. 근데 우리에게 이게 중요한가? (개발파트이신분?)

Monolithic의 문제점

이것이 우리에게 중요한 것!(운영/관리 입장)

- 스케일링이 힘들
 - 기능마다 필요한 리소스 종류와 용량이 다름
 - 인프라 유연성, 활용성이 떨어짐
 - 비용 효율화 급감
- 운영과 관리가 힘들
 - "문제 생겼다. 누구의 문제이지!?"
 - 높은 확률로 자동화가 안되어있음
 - 자동화의 가성비가 없어서
 - "한달에 1~2번 빌드에 무슨 자동화야~ 사람 써!"

"새 업데이트가 있으면 10개 넘는 서버를 직접 수동으로 업그레이드 시켜야해요"

"서버에 문제가 생기면 일일이 다시 복구시켜주세요"

"워크로드를 AWS로 옮기면 오히려 가격이 비싸요"

비슷한 문제를 경험하신 적 있으신가요?

**이런 문제를 해결하고자 하는 고민속에서
Microservices가 언급되기 시작하였습니다.**

(Microservices는 예전부터 논문으로 제시되었던 아키텍처입니다)

새로운 아키텍처 - Microservices

Microservices 도입의 배경

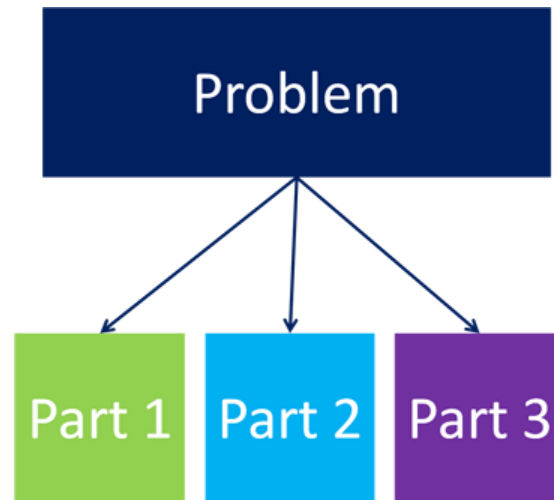
- 빠르게 대응하려는, 혁신을 원하는 서비스 업체에서 시작
 - SI, 치고빠지는 프로젝트형과는 거리가 멀 수도 있음
- 모놀리식은 개발 속도가 너무 느려요.
 - 빠른 세상의 변화, 느린 개발 사이클
 - 코드 결합도가 너무 강해요
 - 기능 개발이 점점 더 복잡해짐
- 큰 서비스를 만들기 힘들어요
 - 큰 CPU, 큰 OS를 모놀리식으로 못만드는거랑 비슷

그러면 모놀리식을 쪼개자(!?)

새로운 아키텍처 - Microservices

쪼개서 문제를 해결하자

Computer Engineering에서는 흔한 문제 해결 방식
(분할 정복 알고리즘-Devide and Conquer)



-> 거대한 모놀리식 서비스를 잘게 잘라진 마이크로서비스로

새로운 아키텍처 - Microservices

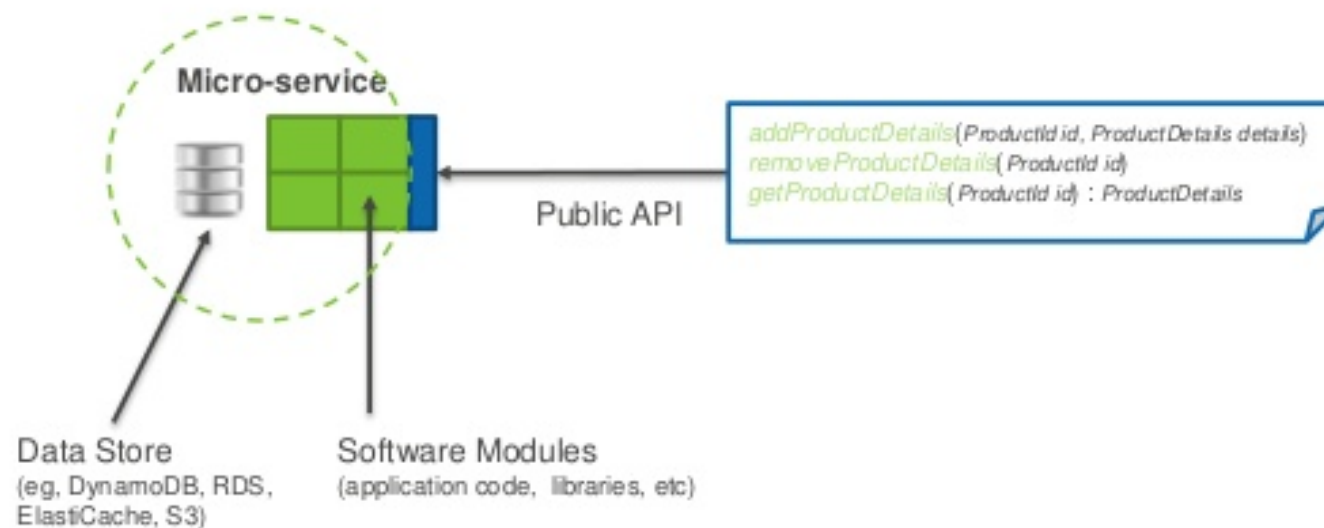
마이크로서비스의 정의

"복잡한 어플리케이션이 **작고, 독립적인** 프로세스들로 이루어진 소프트웨어 아키텍처.

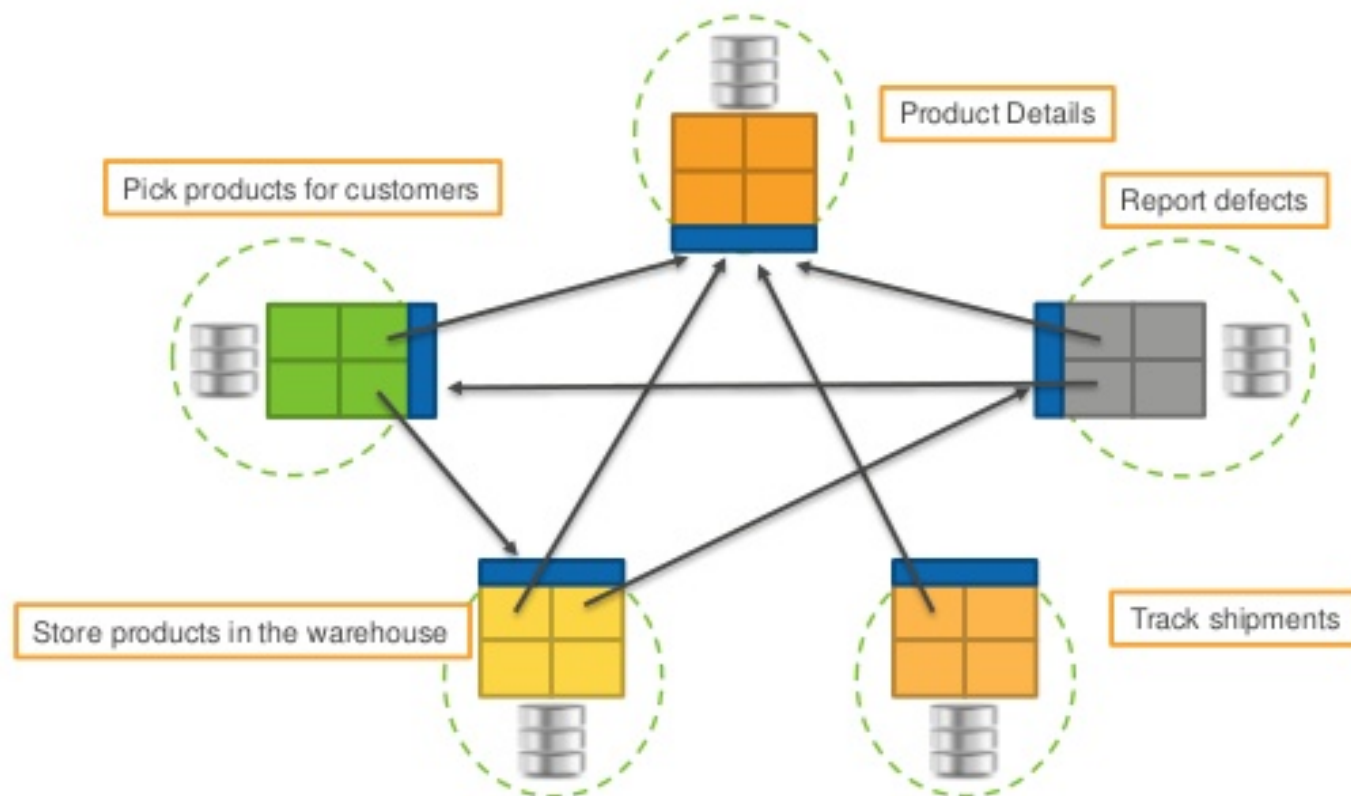
이 프로세스(마이크로서비스)들은 서로에 대해 **매우 낮은 결합도**를 가지고 **하나의 작은 작업**에 집중한다. 이로서 시스템 개발에 모듈화된 접근을 가능하게 한다."

The Anatomy of a Microservice

Micro-service = Fine-grained Service-oriented architecture + “small” public API



Web of Microservices



새로운 아키텍처 - Microservices

마이크로서비스의 정의

- 분산되어 있는
- 독립적인
- 하나의 일을 잘하는
- 적합한 도구(언어)를 골라 사용하는
- 내부를 몰라도 상호작용할 수 있는
- 한 팀이 직접 개발과 운영을 동시에 하는

새로운 아키텍처 - Microservices

모놀리식 vs 마이크로서비스

모놀리식	마이크로서비스
중앙집중	분산
1 프로세스 - N 서비스	1 프로세스 - 1 서비스
1 언어, 공유 자원	적합한 언어, 독립 자원
에러 = 전체 다운	에러 = 부분 다운
한달 1-2 빌드 사이클	지속적 빌드-배포
비효율 스케일링	효율적 스케일링

정리하면

“Do one thing, and do it well”



"Swiss Army" by by Jim Pennucci. No alterations other than cropping. <https://www.flickr.com/photos/pennuja/5363518281/>
Image used with permissions under Creative Commons license 2.0, Attribution Generic license (<https://creativecommons.org/licenses/by/2.0/>)

“Do one thing, and do it well”

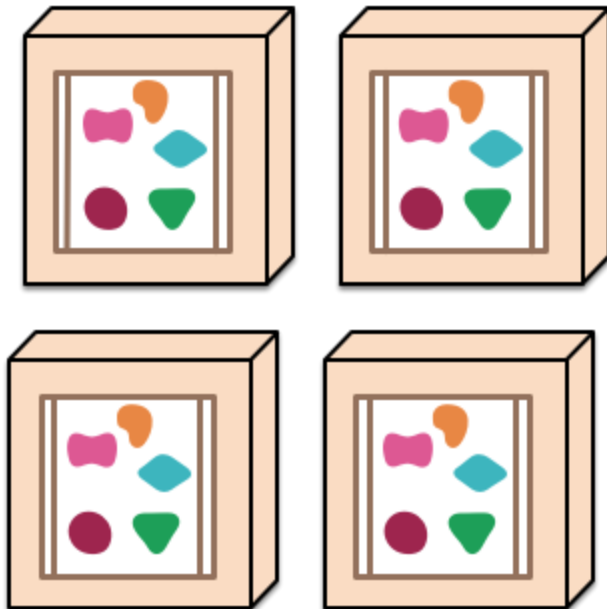


"Tools" by Tony Walmsley: No alterations other than cropping. <https://www.flickr.com/photos/twalmsley/6825340663/>
Image used with permissions under Creative Commons license 2.0, Attribution Generic license (<https://creativecommons.org/licenses/by/2.0/>)

A monolithic application puts all its functionality into a single process...



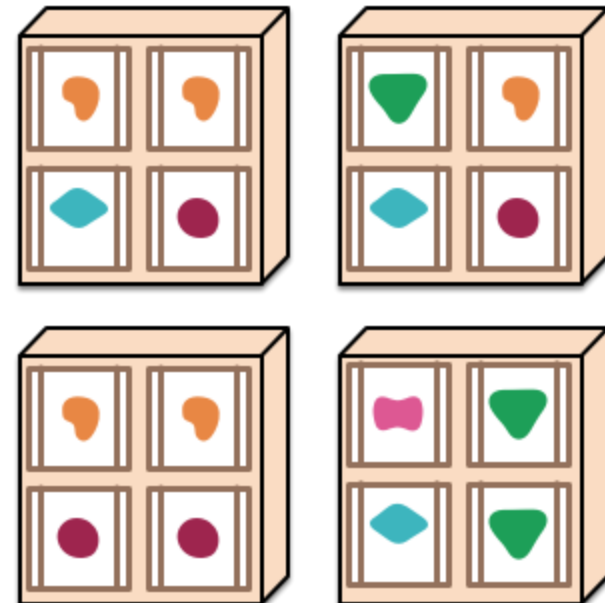
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



정리하면

마이크로서비스 도입하면

개발] 개발은 용이해져요, 쉽지만은 않아요

사업] 많은 실험을 빠르고 리스크 적게 할 수 있어요

설계] 비용, 기능 최적화 가능해요, 복잡하긴 해요

+ 이렇게 안하면 큰 서비스 못해요

...

운영] 일이 많아져요!! - (마이크로서비스"만" 도입했을 때)

왜 운영 일이 많아지는가?

작은 서비스

=> 빠른 개발속도

=> 많은 빌드, 빨라진 빌드(작아진 소스코드)

=> 많은 테스트

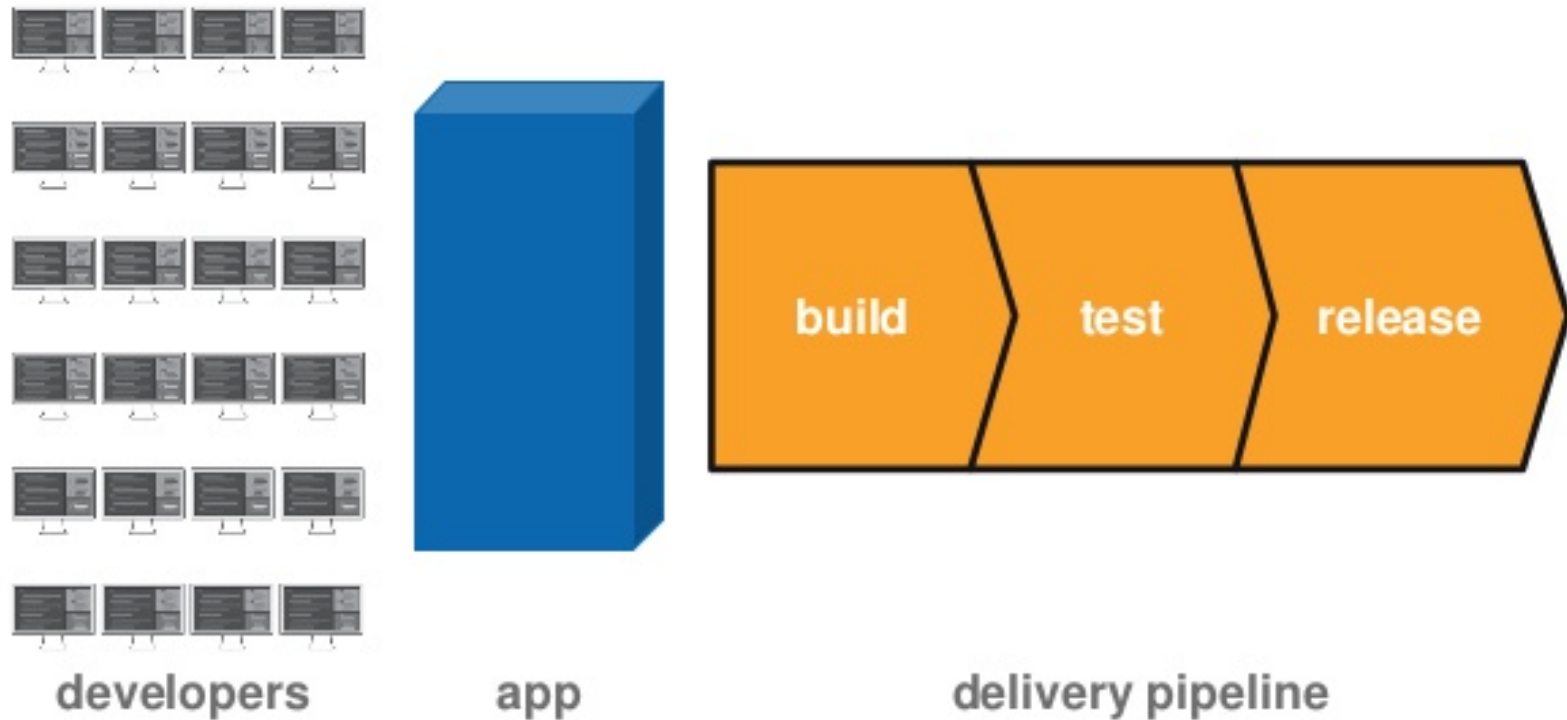
=> 많은 배포

=> 많은 업무량(?!)

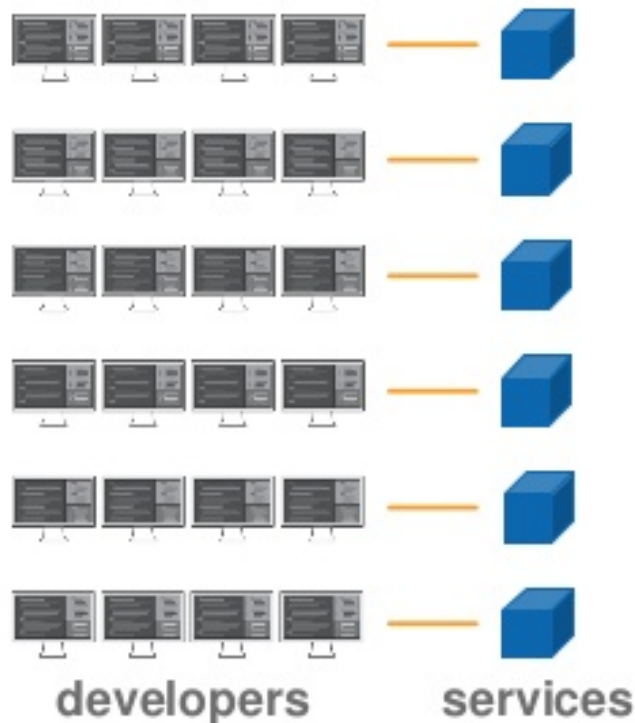
늘어난 프로세스 숫자, 다양해진 배포환경, etc...

안그래도 뻥샀는데, 업무량이 더 늘어나는거 아닙니까?

Monolith development lifecycle



**We found out that we had a tooling gap –
Missing Tools**

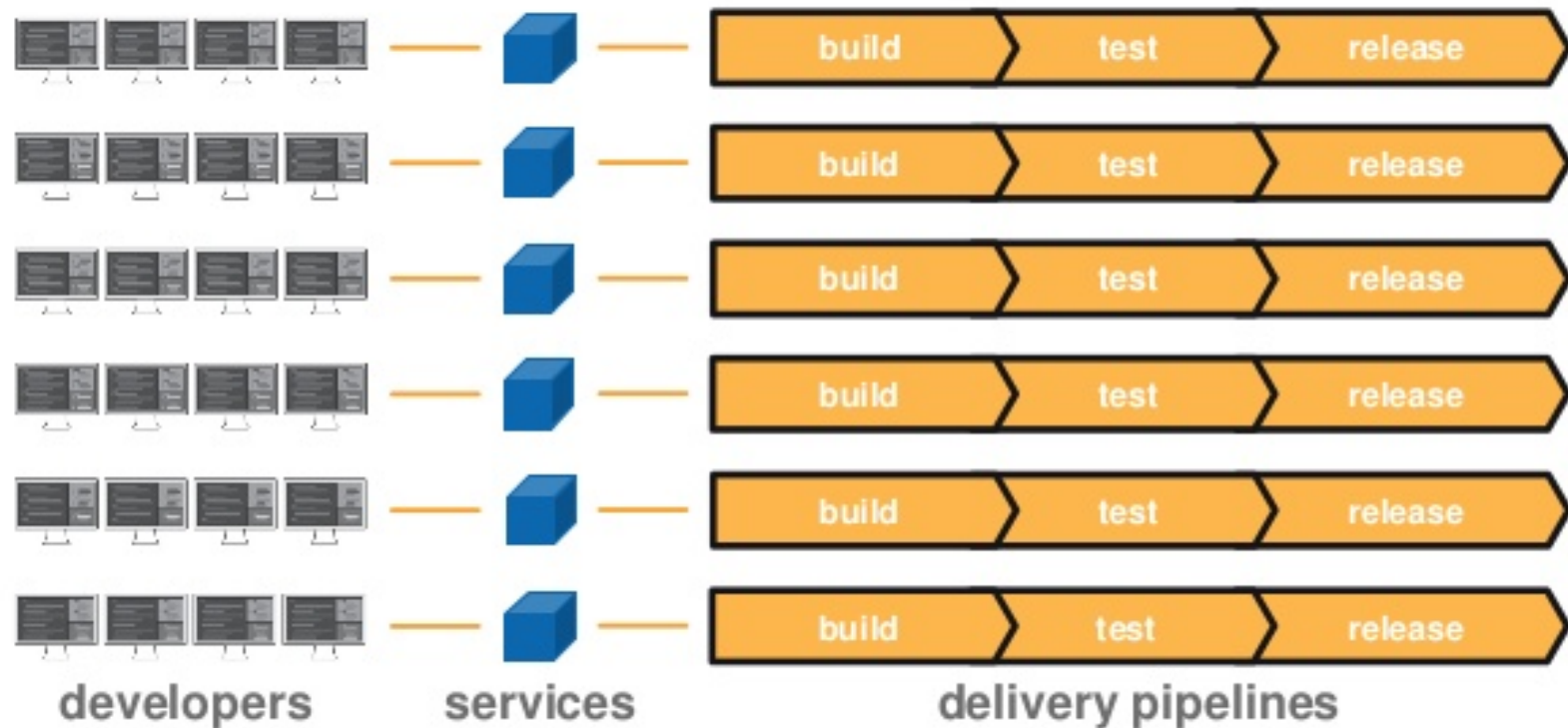


**So....We decided to built
tools to automate our
software release process**



delivery pipeline

Microservice development lifecycle



자동화의 필요성

In 2014 Thousands of teams
× Microservice architecture
× Continuous delivery
× Multiple environments

= 50 million deployments a year

그전까진 어떻게든 사람을 갈아 넣으면(?) 됐지만
이제는 엄청난 양의 배포가 쏟아짐

-> **Delivery Pipeline의 자동화**

배포 문화의 변화

- 일반적인 배포 --> Continuous Delivery
 - 계속 전달, 전달, 전달
- 지속적인 전달을 위해 자동화
 - 테스트 자동화
 - 배포 프로세스 자동화
- 좋은 자동화 도구들이 많이 나옴

Pipeline 자동화, 실질적인 업무량은 줄어듬

Delivery Pipeline 말고도

- 다른 운영 툴들도 자동화, 최적화
- 운영 업무의 양이 사람의 힘으로는 커버 불가
 - 사람이 하더라도 100배, 10000배 더 편해진 도구 발전
- 자동화는 거대한 서비스 유지 필수요소
- (인건비가 비싼 나라에서만 맞는 이야기인가..?)

반대로 말하면

운영 조직의 의미가 사라지기 시작함

- 다 자동화 되니...
 - Netflix의 사례
- 하지만 운영 조직이 사라진다는 의미는 아님
 - 작아져 꼭 필요한 Ops만 두거나,
 - DevOps로 (-DevOps란 무엇인가요?)

Q. MSA에선 운영과 관리는 어떻게 변하나요?

Q. MSA에선 운영관리는 어떻게 변하나요?

A. 조직의 변화를 먼저 이해해야 합니다.

지난 슬라이드에

새로운 아키텍처 - Microservices

마이크로서비스의 정의

- 분산되어 있는
- 독립적인
- 하나의 일을 잘하는
- 적합한 도구(언어)를 골라 사용하는
- 내부를 몰라도 상호작용할 수 있는
- **한 팀이 직접 개발과 운영을 동시에 하는 <-- 여기를 보면**

직접 개발과 운영을 동시에 (DevOps)

- 인프라, 운영툴을 자동화, 추상화

- > 개발자들도 운영에 참여

그렇다고,

개발자들이 운영을 전부 한다는 이야기가 아님

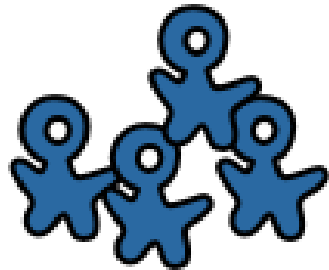
Conway의 법칙

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

-- Melvyn Conway, 1967

요약 : 그 팀에 그 아키텍처

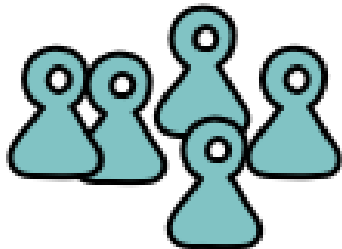
UI
specialists



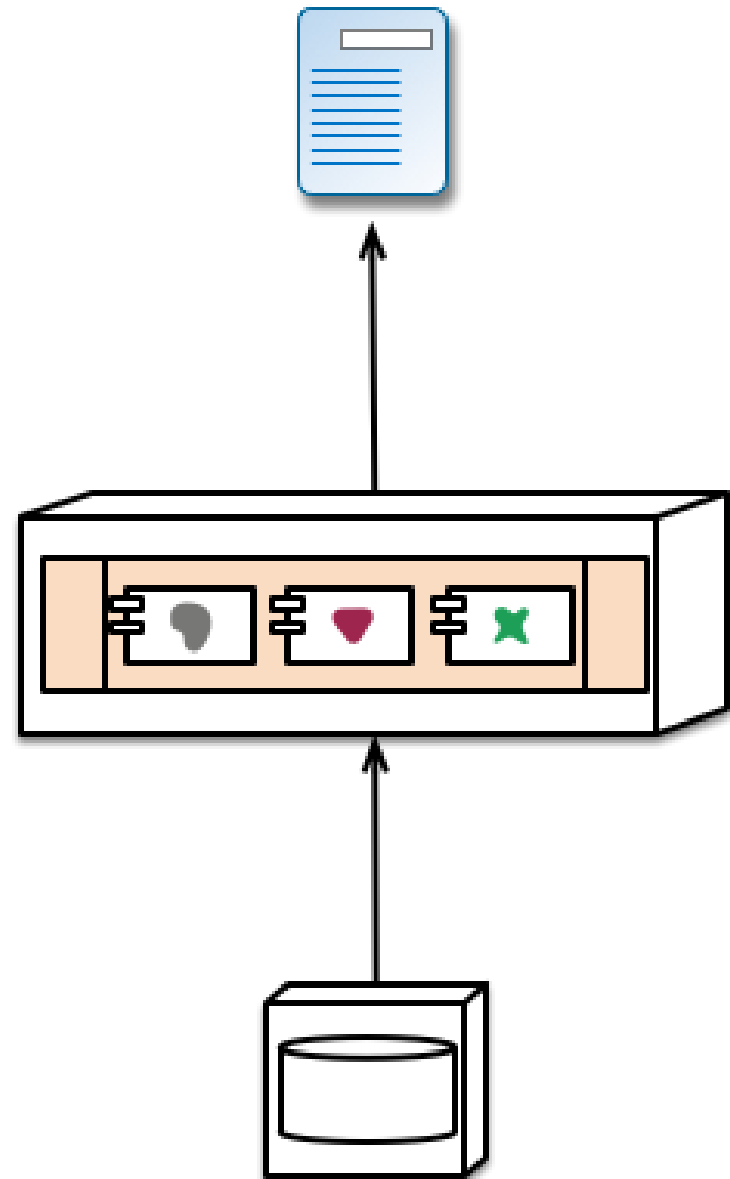
middleware
specialists



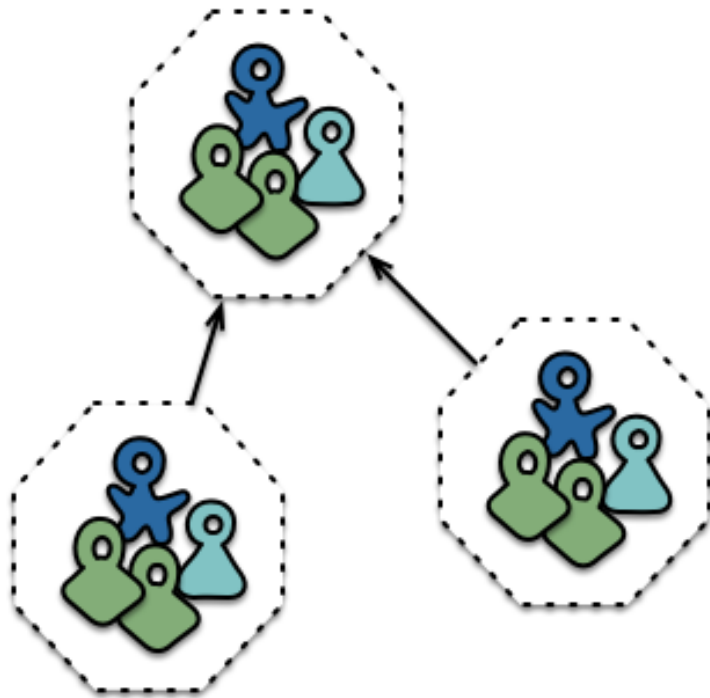
DBAs



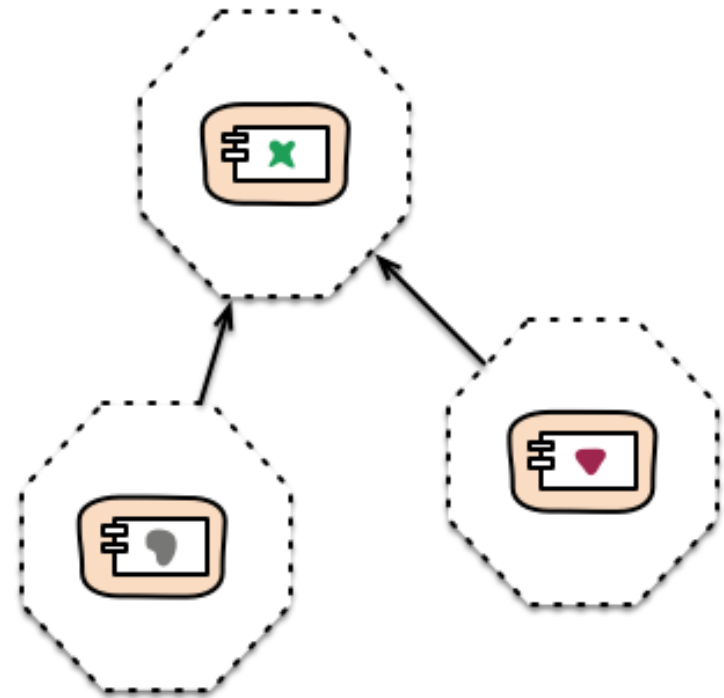
Siloed functional teams...



... lead to silod application architectures.
Because Conway's Law



Cross-functional teams...



... organised around capabilities
Because Conway's Law

아키텍처의 변화 => 조직의 변화

(혹은 조직의 변화 => 아키텍처의 변화)

- 기존과 다른 모양새의 조직이 나타나게 됨
- 기존의 운영/관리 조직 또한 바뀔 것입니다.
 - (개발자들이 운영을 전부 한다는 이야기가 아니라)
 - MSA의 작은 팀들 안에 운영 전문가로서 참여하거나
 - 운영을 좀 더 잘 아는 개발자로서 참여하거나

계속 필요한 운영 Role

- Public Cloud 지식이 풍부한 사람
 - 더 효과적인 '선택'을 위해
 - 레고처럼 조합하는 세상
 - 어떤 블록이 좋은 블록인가? 대답할 수 있는 사람
- 자동화 시스템 개발에 관심있는 사람
 - 비교적 난이도가 낮은 개발
 - 무언가를 만드는 일에 관심이 없다면 비추천
 - 우리가 더 좋은 업무 환경에서 일할 수 있게 만드는 것에 보람을 느낀다면 추천

etc...

변화의 흐름에 탑승하려고 한다면

- "관심"
 - Computer Engineering
 - 프로그래밍
 - 기술

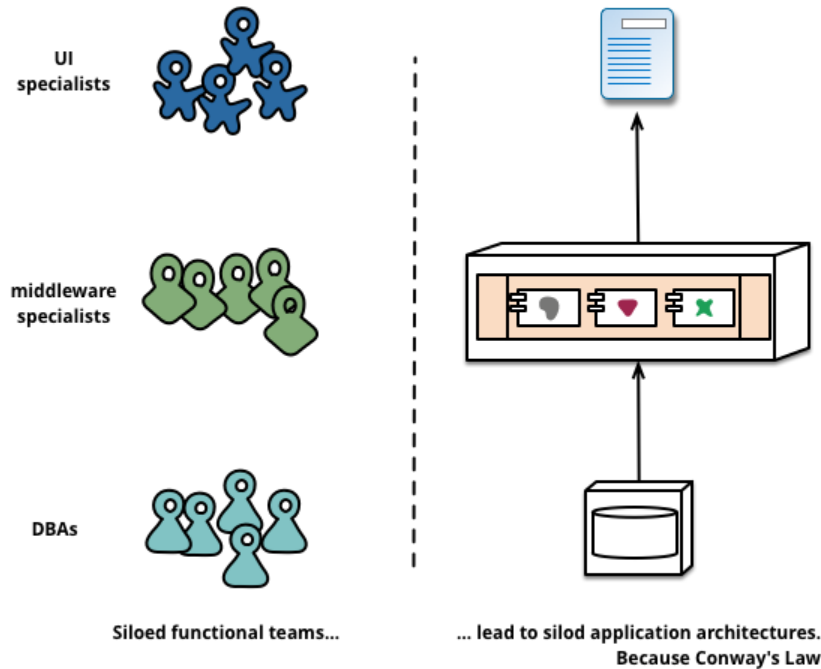
제가 이 교육에서 하고 싶은 것은?

- 여러분들이 관심 가지게 동기부여
 - 이래야 기술적인 이야기가 귀에 들어오지 않을까?

다시 Microservices로 돌아가서..

조직의 차이점이 주는 특징

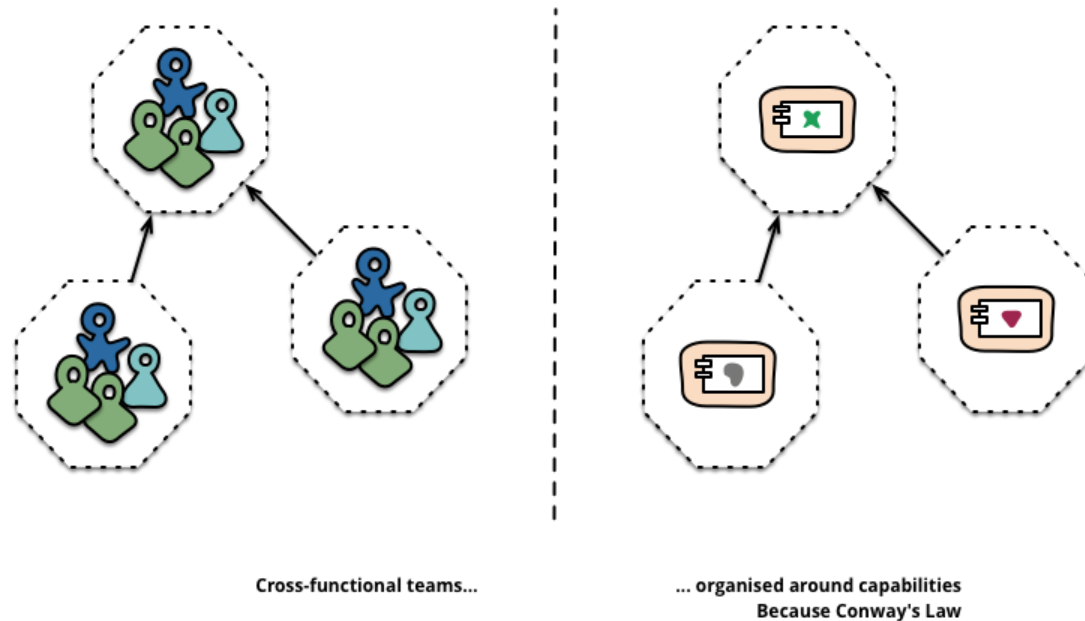
Monolithic vs MSA 조직 구조



특징

- 팀간 낮은 이해도 & 전체를 이해하기 쉽지 않음
- 책임 회피 및 전가 하기에 바쁜...

Monolithic vs MSA 조직 구조



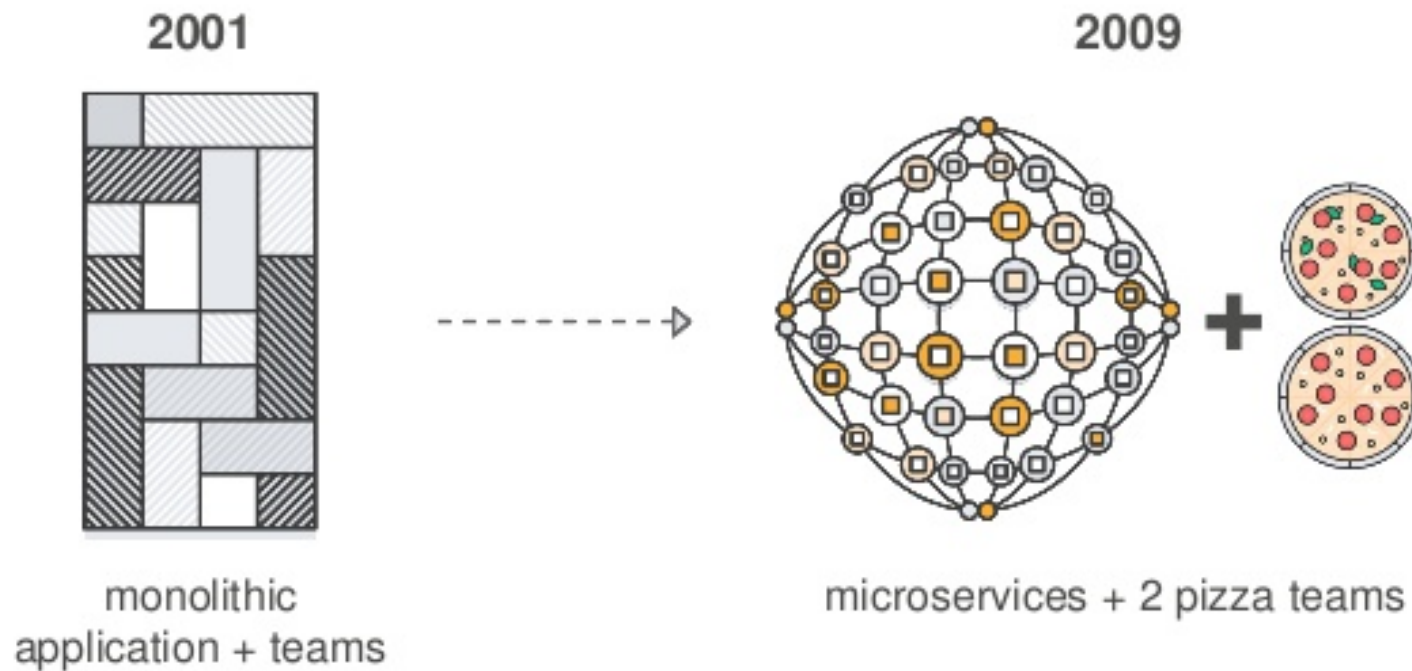
특징

- 작은 서비스 집중 = 작고 확실한 목표 = 높은 업무 이해도
- Q. 그럼 한 팀에 사람은 어느정도가 적합한가요?

AWS와는 무슨 관계인가요?

Amazon.com

Development transformation at Amazon: 2001-2009



AWS is Microservices Architecture



AWS가 MSA에 적합한 것은 우연이 아님

AWS has several characteristics that directly address the most important challenges of microservices architectures

- On-demand resources
- Experiment with low cost and risk
- Programmability
- Infrastructure as code
- Continuous delivery
- Managed services
- Service orientation
- Polyglot

MSA를 써야 AWS를 제대로 쓴다

- Monolithic은 애초에 한계가 있다
 - Trade-off 감안?
- Cost Optimization
- 자동화의 세계에 옛날 방식으로 접근하지 마세요.

정리

- Monolithic의 불편함 --> MSA로 해결
- MSA도 불편함 있음 --> 여러 도구로 해결 (AWS, CI/CD)
 - 가장 중요한 것은 조직 문화의 변화로 해결에 도전
- MSA 잘 쓰려면 AWS, AWS 잘 쓰려면 MSA
- 운영 작아지거나 흡수되거나 사라진다
 - 자동화, 효율화, 조직의 변화
- 새로운 인재상
 - 개발 할 줄 아는 운영 (최소한 관심이라도)
 - AWS 잘 알면 금상첨화

이제야 기술적인 이야기를 할 수 있습니다.

- 처음부터 기술 얘기해봤자...
 - '나랑 관계 없는 일인데?'
- 하지만 지금은?
 - '저걸 아는게 힘이구나'
 - '저걸 왜 알아야 하는지 알겠다!'
 - 동기부여는 최고로 중요

앞으로 할 교육들

- 개발, 사업, 아키텍처링 관점의 MSA
- Continuous Integration / Delivery
- AWS Specific한 이야기들
 - Storage의 종류와 특징들
 - EC2 Instances
- Well-Architected Framework
- Domain Driven Design

앞으로 진행할 교육들

"언어(파이썬) 교육 받고 와도 다음 단계로 공부할 것 찾기가 어려워서 실력이 안늘어 업무에 사용할 자신이 없어요."

- 실습 위주의 AWS 인프라 운용 교육 (코딩위주)
 - Lambda, Cloudformation
 - "이런 Architecture의 Cloudformation Template를 만드시오"
 - "매일 05시에 EC2를 백업하는 자동화를 Lambda 코드를 이용하여 만드시오"
 - 과제(숙제) -> 채점(피드백) -> 모범답안(강의)

단계별 학습 + AWS 실습위주 = 업무 연관성 Up

정리하며...

변화 해야하나?

- 세상은 변하고 있습니다.
- 자동화, 간편화 되어가고 있습니다.
 - 테스트, 배포 프로세스,
 - 인프라 운영도 AWS가
- 개발자들이 운영/관리 할 수 있는 세상이 오는 중
- 선택은 개인의 몫이라고 생각합니다.

변화를 원한다면

- 기술 도입이 해결해주는게 아닙니다.
- 조직이 변해야 합니다.
- 조직이 변하려면 조직원들이 변해야합니다.
- 같은 가치, 같은 목표를 공유해야 합니다.

조직은 철학, 가치, 목표를 공유
조직원은 새로운 기술, 능력을 익히고

여러 부분에서 관심을 가져야
진정한 혁신 가능

피부에 와닿게 말하면,

조직은 더 나은, 행복한 업무생활을 같이 고민,
조직원은 자신을 가치있게 해주는 기술을 익혀

**지속적으로 우리에게 관심을 가져야
좀 더 나은 삶을 살 수 있는게 아닐까...**

감사합니다!

클라우드 서비스팀 남세현