BESPIN GLOBAL

ENTERPRISE CLOUD ADOPTION STRATEGY

존 리(John Lee)는 미국에서 오라클, 마이크로소프트, 세일즈포스, MedeAnalytics, Allscripts 등 다수의 글로벌 기업에서 CIO, CTO 등 임원으로 역임했습니다. 30년 이상의 엔터프라이즈 IT 경험을 바탕으로 베스핀글로벌에서 수석 클라우드 아키텍트이자 글로벌 시니어 기술 임원을 맡고 있습니다.

엔터프라이즈의 클라우드 도입 전략

클라우드를 도입하는 이유는 많지만, 사전에 반드시 "왜?"라는 질문을 해야 합니다. 저는 여러 회사에서 사람들이 이 질문을 하거나, 거부감을 표현하는 행동으로 드러내는 것을 직접 보았습니다. 합당한 우려를 하는 이들도 있지만, 변화를 싫어하거나 직업 안정성이 위협받는 걸 싫어하는 경우도 있습니다. 어떤 경우든, 이들은 클라우드로의 전환을 더디게 하는 장애물의 역할을 합니다.

이 질문에 대한 일반적인 대답은 아래와 같습니다:

- 비용절감
- IT 인프라가 아닌 핵심 역량에 집중하기 위해
- 더 나은 보안과 컴플라이언스
- 사업 성장을 도와주는 다양한 기술 선택지
- 경쟁사가 클라우드로 이동함

이 대답이 개발자, IT 매니저 등 기술 직군에게 확신을 주지 못할 수도 있습니다. 이들의 눈에는 경영진이 이해득실과 예상되는 어려움은 이해하지 못한 채 또 그럴듯한 무언가를 쫓아가는 것처럼 보일 수 있습니다.

뻔한 대답이 아니라, 사업 특성에 따른 구체적인 이유와 마이그레이션, 아키텍처에 대한 전략적인 로드맵을 준비해야 합니다.

클라우드 도입 계획하기

클라우드 도입 성공의 핵심은 초기에 빠르게 성공하는 것입니다. 초기 프로젝트는 작고 애자일하며, 비즈니스 크리티컬하지 않아야 합니다. 기술 직군이 배우면서 확신을 얻을 수 있을 수 있는 PoC 정도로 생각할 수 있습니다. 이 과정을 통해 나중에는 이들이 클라우드 도입 에반젤리스트가 될 수 있습니다.

초기 프로젝트 대상으로는 HTTP/2 프로토콜을 사용한 REST API를 통해 다른 서비스에 접근하는 기존 온-프레미스 솔루션을 선택하는 것이 좋습니다. GET, PUT, POST, DELETE와 같은 제한된 메소드를 사용하는 선언적 API 명세를 사용하여 신규 클라우드 네이티브 기능을 추가해야 합니다.

이와 함께 또는 이를 대체할 방법으로, 전사에서 사용되는 소프트웨어보다는 회사 내 특정 그룹만 사용하는 온프레미스 솔루션으로 시작하는 것입니다. 잘 정의된 워크플로우와 논리적인 구성의 서비스 지향 아키텍처(SOA)를 가진 소프트웨어를 선택하는 게 좋습니다.

상품 정보 조회 서비스와 같이 무 상태(stateless), 읽기 전용(read-only) 비동기 서비스로 시작해야 합니다. 이런 경우 콜백(웹 훅) 처리를 통해 서비스 풀에 속한 다른 인스턴스가 요청 처리 및 재시도를 할 수 있도록 기존 코드를 리팩토링해야 할 수도 있습니다.

어떤 유형의 솔루션을 선택하더라도 모두 페일 오버 및 성능을 위해 고가용성(HA) DB가 필요합니다. 또한, DB는 스키마 테이블 레벨 명령형(imperative) 질의를 사용하는 것보다는 메타-선언형(meta-declarative) 요청을 지원해야 합니다. 이것은 데이터가 서비스에 복사되는 방식인지 중앙 DaaS(Data as a Service)의 추상화를 사용하는지 여부와 관계없이 요구됩니다.

클라우드 마이그레이션 과제 시작하기

저희 회사에서는 클라우드 도입을 위해 크게 네 가지(PPPT, 철학, 사람, 프로세스, 툴)가 필요하다고 정의합니다. 이 네 가지를 조합하면 '왜?'에 대한 대답이 나옵니다. 그 대답은 회사 오너부터 기술자까지 서로 다 다를 수 있습니다.

철학은 비전입니다. CIO의 비전과 개발자의 비전은 매우 다를 수 있습니다. 그러나 그 비전의 핵심 가치는 같습니다. 바로 비즈니스 가치를 창출하고, 구성원 개개인을 위한 가치를 창출하는 것입니다.

간과하기 쉬운 가장 큰 과제는 사람입니다. 특히 기술을 판매하는, 그래서 핵심 자산이 사람인 회사는 더 그렇습니다. 개인적인 경험으로 볼 때, 대부분의 기술자들은 새로운 기술을 배우고 새로운 시도를 하는 데 열려 있습니다. 그럼에도 변화를 거부하거나 변화하지 못하는 사람은 어디에나 있습니다. '왜', '어떻게', 심지어 '누가'에 대한 구체적인 대답이 나와도 말입니다.

이들을 무시한 채 의지가 있는 직원들만 데리고 도입을 진행할 수도 있습니다. 하지만 협조적이지 않은 이들이 결국 걸림돌이 될 것입니다. 그래서 저는 과제에 착수하기 전에, 보통 전체 기술 직군을 모아놓고 아래와 비슷한 말을 합니다.

> "우리는 이익과 어려움을 모두 분명하게 인지한 상태로 클라우드로의 여정을 시작하려고 합니다. 여러분 개개인, 그리고 회사의 성공을 위해 필요한 지원과 교육을 모두 제공할 것입니다. 이번 과제는 여러분을 회사에 더 중요한 인재로 만들고, 여러분의 커리어도 발전시킬 것입니다. 양쪽의 공동 성공을 위해 회사는 투자를 아끼지 않을 것입니다….

여러분 중 일부는 현재 가진 스킬에 많은 투자를 했고, 변화를 원하지 않는다는 것을 알고 있습니다. 여러분이 주저하는 것도 이해를 합니다. 동시에, 여러분의 지원을 필요로 하는 고객들이 여전히 있습니다. 여러분은 여전히 회사에 중요한 존재이며, 기여한 부분에 대해 인정받을 수 있게 할 것입니다. 하지만 우리의 계획은 신기술을 가진 직원을 더 많이 채용하는 것이며, 앞으로 몇 년간 현재 기술 스택은 완전히 바뀔 것입니다.

이러한 변화에 동참하고 싶지 않다면, 그 뜻을 충분히 존중하며, 회사를 떠나는 절차가 최대한 매끄럽도록 지원할 것입니다…."

이 말을 하면서 저는 기술 직군의 5% 정도를 잃을 각오를 합니다. 하지만 평균 연간 이직률이 10-15%인 것을 감안하면, 이러한 조치는 전략에 동의하지 않는 일부 사람들을 내보내고 성과가 좋은 엔지니어들이 계속 남아 성장하도록 하는 순기능을 합니다.

클라우드 네이티브한 개발 (CND)

기존의 모놀리식 소프트웨어 개발 프로세스와 툴은 개발 영역과 운영 영역이 분명하게 구분됩니다. 개발자들은 요구사항을 도입하고, QA 테스트 팀은 이들과 분리돼 있기 때문에 여러 종류의 배포 모델을 이해하지 못합니다.

소프트웨어가 출시되고 나면 운영팀이 이를 넘겨받아 배포가 제대로 됐는지 확인하고, 확장성과 성능을 모니터링합니다. 이들이 HA와 DR을 수립할 때는 개발자들과 전혀 다른 프로세스와 툴을 사용합니다.

클라우드 네이티브한 개발 패러다임은 모놀리식 프로세스를 완전히 바꿉니다.

클라우드 네이티브 개발에서는, 배포, 성능, 확장성, 고가용성, 재해복구 같은 운영 요구사항은 소프트웨어 아키텍처의 첫 번째 고려사항입니다.

뿐만 아니라, CND 배포가 하나의 클라우드 벤더에서만 이뤄질 거라는 보장은 없습니다. 엔터프라이즈 CND 솔루션은 여러 개의 클라우드 벤더에 걸쳐 있을 가능성이 높습니다. 이것은 인수합병의 결과일 수도 있고, 팀들이 스스로 클라우드 벤더를 선택한 결과일 수도 있습니다. 두 상황의 조합일 가능성도 높습니다.

이것을 염두에 뒀을 때, 첫 번째 요구사항은 데브옵스/클라우드옵스입니다. 카나리, 블루-그린, 일부 엔터프라이스 CND의 경우에는 레드-블랙과 같은 모델에서 소프트웨어의 상태 없는 배포가 가능해야 합니다.

더 조밀한 서비스 지향 아키텍처(SOA)와 더불어, 개발자들은 성능과 확장성을 프로그래밍 구성 단계에서 고려해야 합니다. 모든 것이 유연한 서버리스 클라우드 서비스의 시대가 되면서, 인프라에 코딩하는 능력은 점점 필수가 되고 있습니다(Infrastructure as a code).

이것은 클라우드 네이티브 개발을 위한 여정의 시작일 뿐입니다. 확장성을 위해 리소스를 최대한 활용하려면, 다음 여정은 마이크로서비스입니다. 도커스왐, 컨술(Consul), 쿠버네티스와 같은 컨테이너 관리 시스템이 여기에 사용됩니다.

베어메탈, VM웨어, 하나 혹은 그 이상의 컨테이너를 호스팅하는 쿠버네티스 포드의 배포 횟수와 사이즈, 그리고 클라우드 환경의 개수까지 생각하면, 아키텍처는 다이나믹 서비스 디스커버리와 관리에 의존해야 합니다. 이것은 프로그래밍된 접근 룰 기반으로 정적 서비스에 전송하는 일반 하드웨어 라우터에서는 매우 어렵고 비쌀 수 있습니다.

멀티 클라우드 간 CND 모델에서 서비스 메시는 필요한 모든 IP 기반의 컨테이너 포드들에 TCP 연결을 가능하게 합니다. 마이크로서비스가 원격 프로시저 호출(gRPC)을 통해 컨트롤러에 명령을 내리고, 제어 평면(Control plane)은 REST API (http/2)를 통해 모든 다운스트림 트래픽을 관리해야 하는 환경에서는 중앙 제어 옵션을 고려해야 합니다. 이런 구성은 항상 중앙 코디네이션이 필요하고 엄격한 동기 워크플로우를 요구하는 서비스에 사용할 수 있습니다.

완전히 분산된 마이크로 서비스를 위한 다른 선택으로, 아키텍처는 모든 주요 구성 정보와 마이크로 서비스의 단기 속성을 다루기 위한 보안 관리 목적의 일부 제한된 운영 데이터를 하위 컴포넌트로 전달할 수 있어야 합니다. 이 기술은 엔보이(Envoy) 같은 사이드카를 포함합니다.

두 경우 모두 서킷브레이커와 보안 관리(Secret management)에 대한 요구사항은 고가용성과 보안을 위해 꼭 필요합니다.

서킷 브레이커는 연결중인 서비스가 다운되서 재시도 했을때 다른 서비스로 리디렉션 하는 역할을 합니다. 시크릿 관리(secret management)는 수십 개에서 수천 개의 프라이빗 토큰이 서비스와 사용자에게 분산되어 있을 때 보안 관리를 해줍니다.

이러한 보안 아키텍처는 일정 시간이 지나거나, 특정 행동을 하거나, 요청했을 때 암호를 파기하고, 선택적으로 새로 설정하거나 변경해줄 수 있어야 합니다. 볼트(Vault)와 같은 툴이 서비스 메시 안에서 이를 관리할 수 있게 도와줍니다.

마지막으로, 로그 기록과 오딧 요구사항입니다.

미국 건강보험 양도 및 책임에 대한 법(HIPAA), 미국 건강정보신탁연합(HITRUST), 미국 CPA 협회의 SOC2 감사, 유럽연합의 일반 개인정보 보호규정(GDPR), 그리고 다른 국가 및 지역의 요구사항이 반영돼야 합니다. 이것은 역할 기반 접근 제어(Role-Based Access Control)를 통해 어느 정도 충족할 수 있는 부분입니다. 그러나 접근 룰에 부합하는 데이터 접근이 이뤄졌을 때, 유저의 로그와 쿼리의 내용, 날짜, 시간이 감사를 위해 기록돼야 합니다.

제가 추천하는 것은 감사를 위해 몇 개의 거버넌스 서비스를 조합해서 사용하는 것입니다. 이는 서비스 별로 매우 구체적일 수 있으며, 클라우드 및 온프레미스를 비롯한 모든 데이터드리븐 서비스에 이 요구사항이 적용됩니다.

마이크로 서비스 아키텍처의 접근 요청과 메소드 증가를 고려하면, 아키텍처 설계 단계에서부터 로그가 시계열 스키마로 제대로 기록되도록 설계해야 합니다. 이것은 데이터 거버넌스 관련 알림이 정확한 담당자에게 전달되도록 해줍니다.