Twolinecode

쉽고 빠른 Cloud-Native 기반의 서비스 개발 및 구축

Cloud-Native, MSA

2023.08.23

Contents

01 회사 소개

02 MSA 전환 및 구축 전략

03 How to modernization

04 MSA를 위한 필수 요소

05 투라코(Turaco) 솔루션 소개

01 회사소개

- 회사 개요
- History
- 사업영역

투라인코드 회사 개요

Twolinecode

" 클라우드에 대한 풍부한 경험을 기반으로 고객의 니즈를 정확하게 파악하고솔루션에 반영합니다 "

전문 기술력을 통한 컨설팅 및 개발 운영등 안정적인 DX 전환 지원이 가능합니다.
클라우드 네이티브 개발을 위한 전문 솔루션을 제공 하고 있습니다.

회사명
(주)투라인코드 (Twolinecode Inc.)

대표이사
현승엽

설립일
2013년 3월

인력구성 : 총 82명
개발 : 72명 / 마케팅, UIUX, BX : 7명 / 경영지원 : 3명
국내외 클라우드 전문 자격증 : 56개 이상

Head Office
경기도 성남시 분당구 성남대로 331번길 11-3, (정자동) 여민빌딩 5층

R&D Center
경기도 성남시 분당구 성남대로 343번길 12-2, 신수빌딩 3층

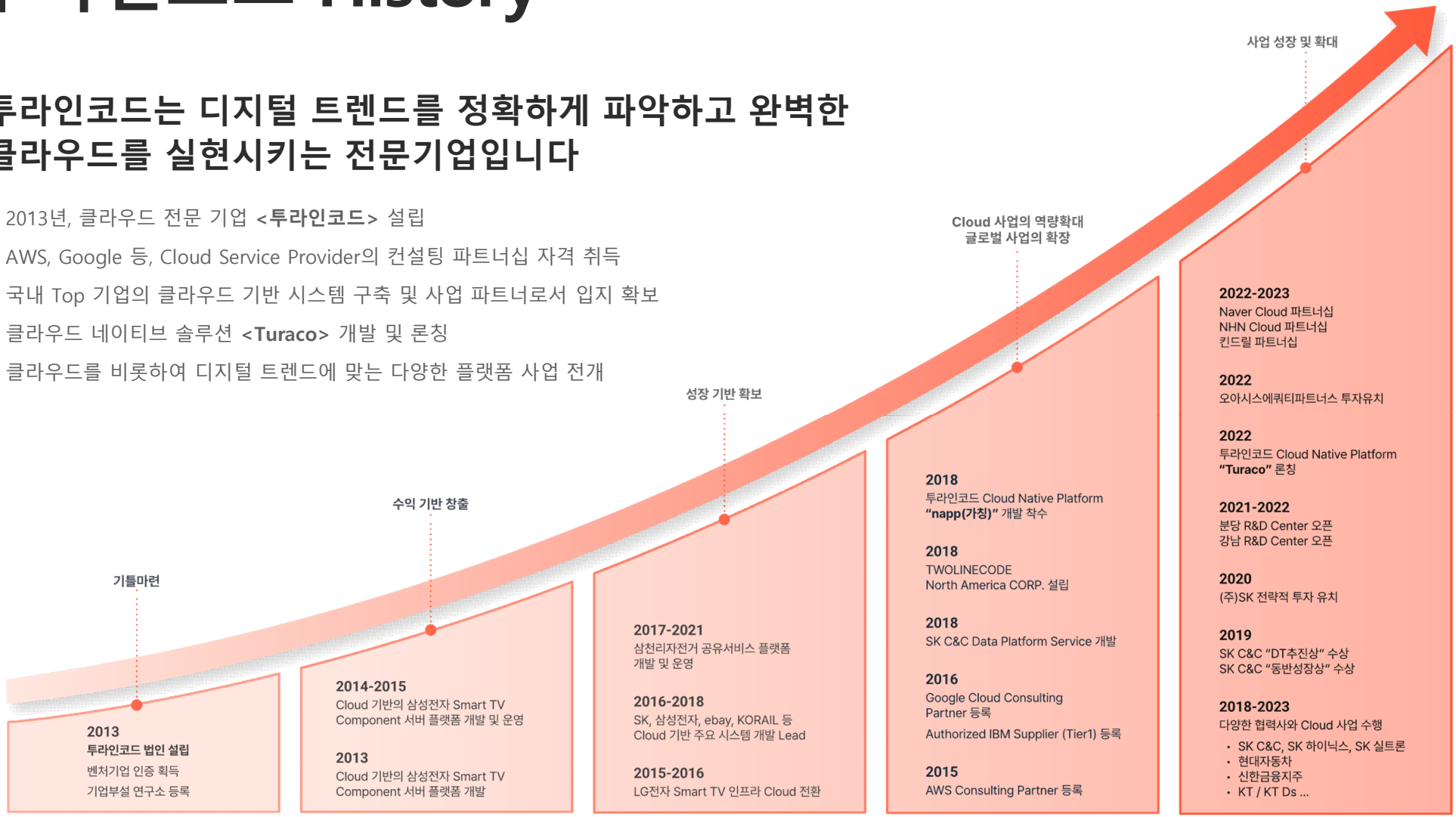
R&D Center
서울시 강남구 테헤란로 2길 27 패스트파이브 강남 5호점 13층

New York Office
127 West 30th St. Unit #1014 New York, NY 10001 USA

투라인코드 History

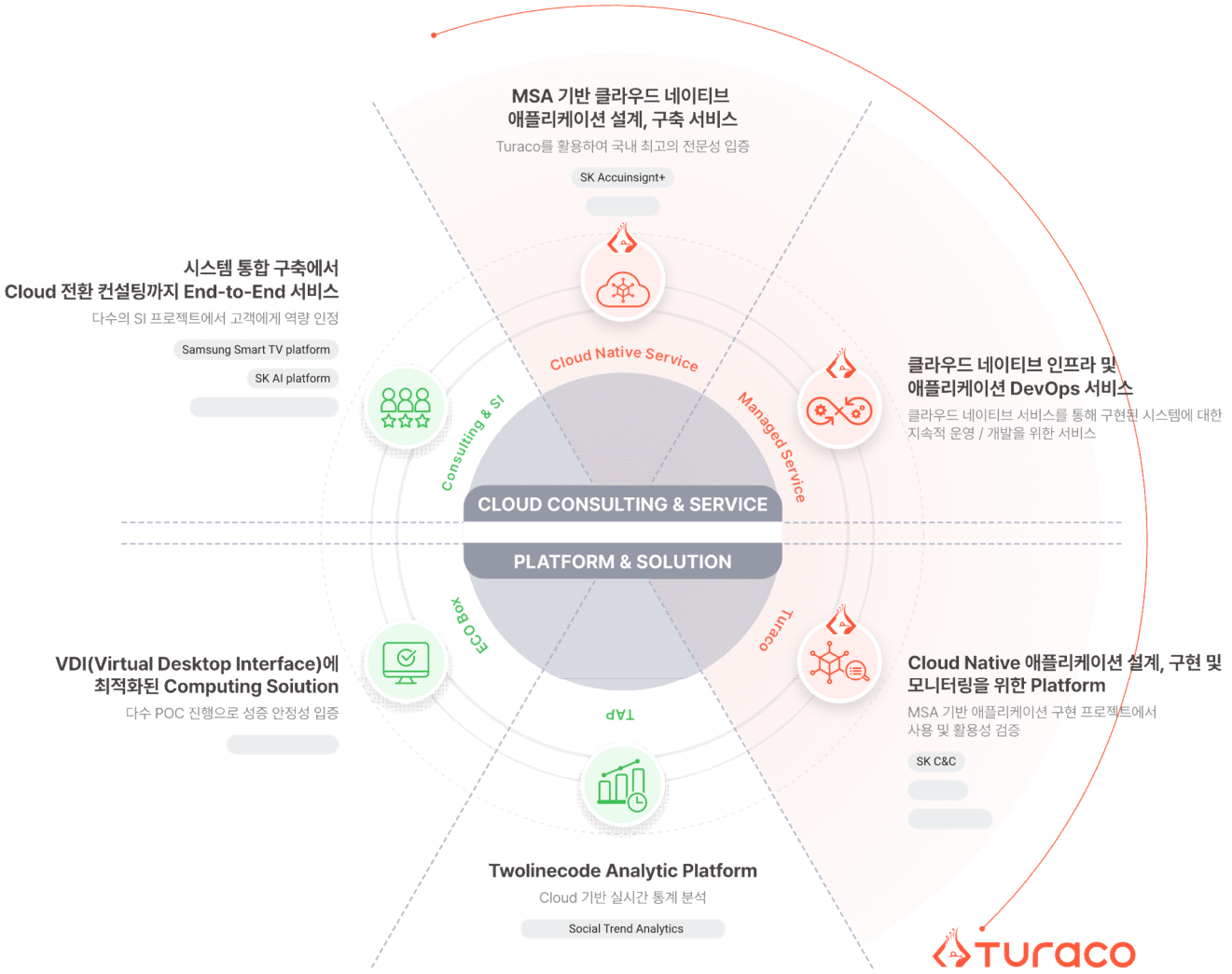
투라인코드는 디지털 트렌드를 정확하게 파악하고 완벽한 클라우드를 실현시키는 전문기업입니다

- 2013년, 클라우드 전문 기업 <**투라인코드**> 설립
- AWS, Google 등, Cloud Service Provider의 컨설팅 파트너십 자격 취득
- 국내 Top 기업의 클라우드 기반 시스템 구축 및 사업 파트너로서 입지 확보
- 클라우드 네이티브 솔루션 <**Turaco**> 개발 및 론칭
- 클라우드를 비롯하여 디지털 트렌드에 맞는 다양한 플랫폼 사업 전개



투라인코드 사업영역

클라우드 컨설팅 / 서비스, 플랫폼/솔루션



02 MSA 전환 및 구축 전략

- Problem?
- What is Cloud-Native?
- Why Cloud-Native?
- Benefits of Cloud-Native
- How to Cloud-Native?

Problem? 기존 조직 및 서비스가 가진 문제들...

✓ 편의성 부족

디지털화 되지 않은 서비스 존재, 디지털 플랫폼에 최적화 되지 않아 서비스 이용에 제한
→ 사용자들의 서비스 이용률 및 만족도 저하

✓ 서비스 현대화 / 느린 업데이트

플랫폼 및 시스템 업데이트, 서비스에 대한 현대화를 제때 수행하지 못해 최신 기술 및 서비스 미적용
→ 편의성 및 효율성이 낮아져 사용자들의 이용률 저하

✓ 서비스 통합

조직마다 독립적인 시스템과 서비스 운영으로 서비스 통합이 어려움
→ 기관별 서비스를 각각 방문하여 정보를 획득해야 하기에 사용자 경험 및 만족도 저하

✓ 복잡한 절차 / 정보 제공 이슈

정보 제공 절차의 복잡성 등으로 인한 서비스 이용 방법의 어려움과 제공 정보 부족
→ 사용자가 필요한 정보를 찾지 못하는 등 서비스 품질 저하

✓ 디지털 역량 부족

조직의 IT 역량 부족으로 새로운 기술의 흐름에 유연하게 대응하지 못함
→ 현대적인 서비스 제공이 불가능하며 DT 지연

✓ 정보 보안 이슈

서비스에서 수집하고 사용되는 개인 정보 및 민감 정보 그리고 시스템 업데이트 이슈로 인한 취약점 발생 가능
→ 개인 정보 및 통신 방식의 암호화 미적용, 시스템 보안 업데이트 미적용으로 인한 해킹 가능성

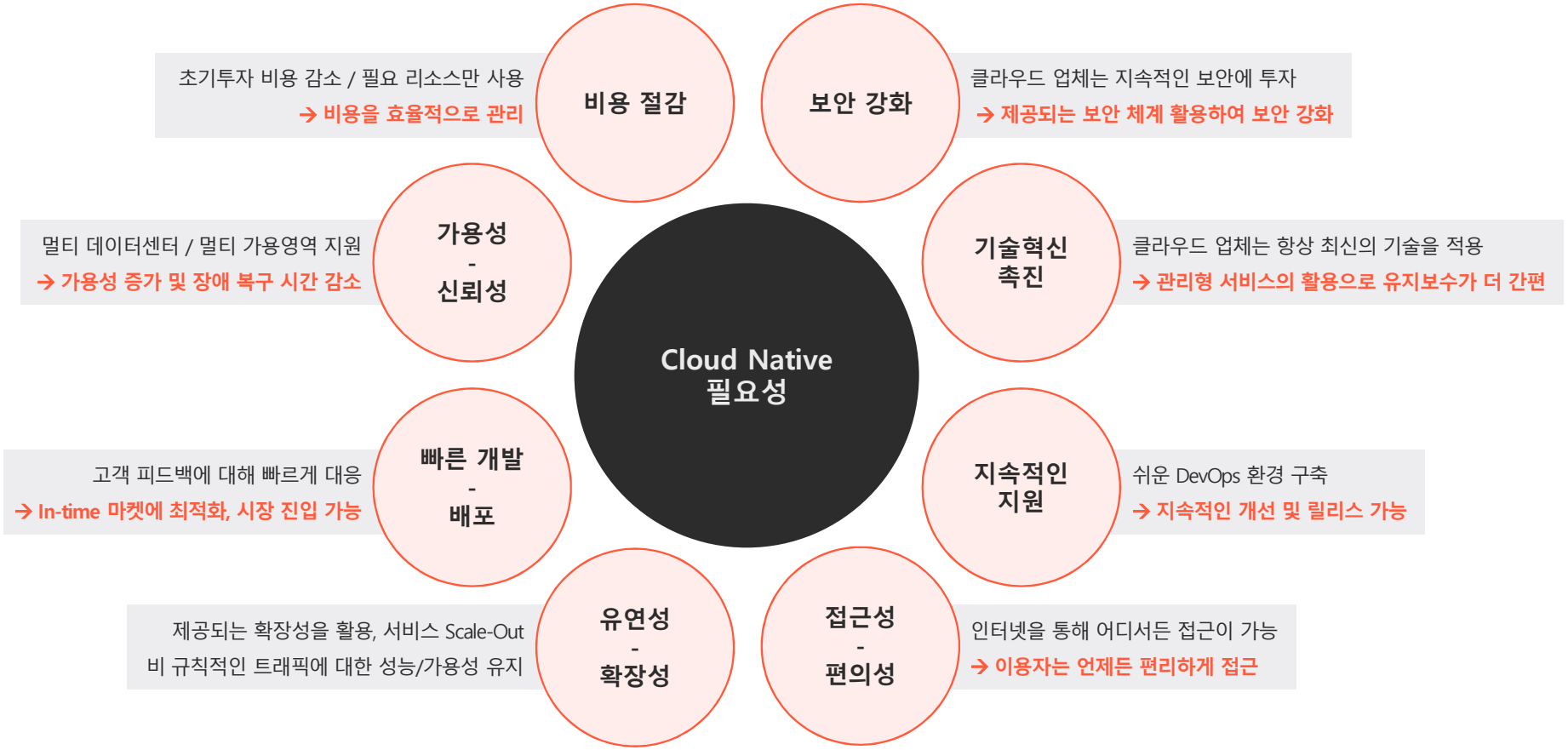
What is Cloud-Native? 클라우드 네이티브란 무엇인가요?

- 클라우드 컴퓨팅 기술을 활용하여 애플리케이션 및 인프라를 관리
- 더욱 빠르게 확장 가능하며 안정적으로 개발, 배포, 운영하는 아키텍처 패러다임
- 시장의 변화에 빠르게 적응하고 사용자에게 혁신적인 서비스를 제공

Cloud Native

컨테이너화 (Containerized)	애플리케이션의 실행 환경 및 종속성을 모두 컨테이너 내에 패키징하여 스테이지별 환경의 일관성 확보 → **애플리케이션의 이식성 및 확장성 향상**
서비스 지향 아키텍처 (Service Oriented Architecture)	도메인별 독립적인 서비스로 분해하여 개발되고 배포되며 개별적으로 운영 → **느슨한 결합도(Loose-coupled)를 가지며 개별 서비스가 지향하는 비즈니스에 집중할 수 있도록 함**
자동화 (Automation)	컨테이너화된 수많은 서비스는 자동으로 각 스테이지별로 실시간 배포되고 관리되어야 함 → **개발 및 배포 시간의 감소로 고객 피드백 소요 시간을 최소로 보장할 수 있음**
가상화된 인프라 (Virtualized Infra)	수많은 Backing 서비스 관리는 효율적이어야 하며 이 또한 코드 형태(IaC)로 관리 → **인프라 변경 및 구축을 빠르게 할 수 있으며 유지 보수 비용도 절감**
확장성 (Scalable)	가변적인 트래픽에 대해 리소스 또한 가변적으로 확장되고 축소되어야 함 → **빠른 응답과 가용성을 높임**
무 상태 아키텍처 (Stateless Architecture)	애플리케이션의 상태를 유지하지 않아야 하는 "무 상태 아키텍처 " 를 가져야 함. → **애플리케이션 간 교체를 쉽게 할 수 있고 배포 및 관리 또한 용이**

Why Cloud-Native? 왜 클라우드 네이티브 기반으로 전환을 하려 하는가?



Benefits of Cloud-Native 전환 시 기대 효과는 무엇인가요?

기술적인 측면

- **기술혁신과 민첩성 증가**
  - 기술 다양성과 독립성
  - 개발 생산성 향상
  - 빠른 배포
  - 장애 격리 및 회복성
  - 서비스 스케일링
  - 쉬운 기술 혁신

검토 사항

- 기술적인 복잡성 검토
- 인프라 변경에 대한 고려 필요
- 전환 전략과 철저한 테스트 필수

비즈니스 측면

- **기업 성장과 경쟁력 강화**
  - 빠른 기능 개발과 출시
  - 유연한 팀 구성 및 독립성
  - 높은 가용성 / 신뢰성
  - 더 나은 고객 경험 제공
  - 성능 향상 및 확장성
  - 비용 절감

검토 사항

- 기업의 현실적인 상황 및 리스크 고려
- 체계적인 전환 전략 및 수행

관리 / 운영 측면

- **운영 효율성**
- **유지 보수 비용 최소화**
  - 각 서비스의 독립적 운영
  - 높은 확장성 / 가용성
  - 배포 관리 용이
  - 회복성과 격리
  - 서비스 모니터링
  - 기술 혁신 촉진

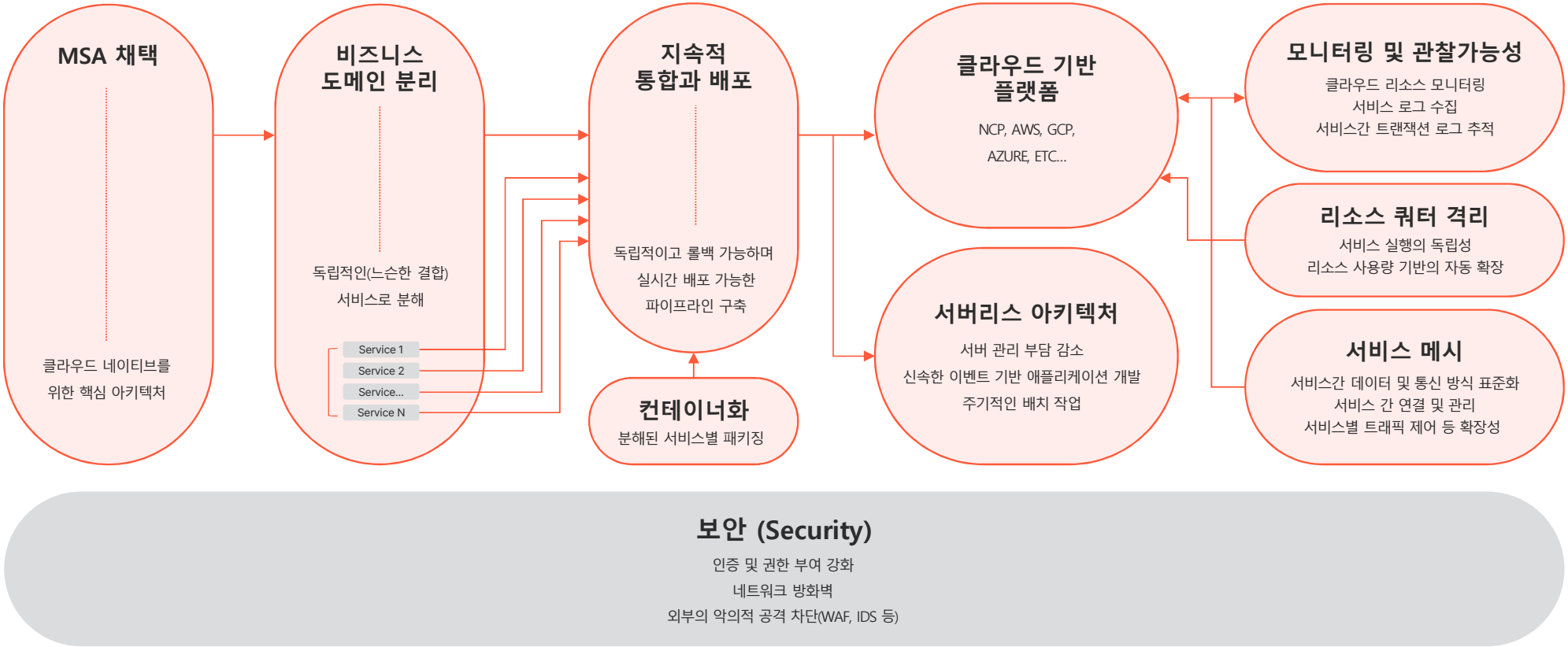
검토 사항

- 조직 내의 변경 관리
- 팀간 협업 조율
- 기술적인 노력이 필수

How to Cloud-Native? 클라우드 네이티브로 전환은 어떻게 하나요?

식별 가능한 비즈니스 도메인의 수, 연동중인 외부 솔루션 종류 및 적용 방식

→ 전환 대상이 되는 시스템의 복잡성과 기술적인 요구 사항에 따라 다양한 기술이 존재



03 How to modernization

- Big-Bang vs Strangler
- 신규 구축
- 마이그레이션
- 데이터 베이스 전환
- 마이크로 서비스 간 데이터 요청
- API 방식 + CDC 방식

How to modernization? Big-Bang vs Strangler

Big-bang

- **일괄적 전환 방식**
  - 모든 기능 및 컴포넌트를 새로운 아키텍처로 교체
  - 현대화된 시스템으로 한번에 전환
- **시스템 중간 단계 유지 불필요**
  - 일시적 호환성 문제 발생 가능성 낮음

단점

- 전체 시스템의 일괄 변경으로 인한 위험성 높음
- 대규모 변경으로 인한 시간 및 비용 소요
- 기간 및 비용이 많이 소요되어 프로젝트 전체 일정에 미치는 영향 높음

Strangler

- **안정적, 점진적 전환 방식**
  - 기존 시스템을 작은 단위로 분리, 부분적으로 현대화 수행
  - 기존 시스템과 전체 시스템에 영향 최소화
  - 일부 기능 대체 → 기능 추가, 수정 용이
  - 기존 시스템과 병렬 개발 가능
  - 비용 분산 전환이 가능하여 초기 비용 최소화

단점

- 부분적인 전환으로 전체 시스템 전환 시간 소요
- 기존 시스템간 호환성 고려
- 서비스 간 기능 중복 및 데이터 불일치

How to modernization? 처음부터 새로 구축하고 싶어요

컨설팅

요구 기능 분석

요구 사항 분석
- 비즈니스 분석
- 전략 개발
- 기술 선택
- 아키텍처 설계
- 비용 산정 및 최적화
- 보안 요소 결정
- 모니터링 거버넌스
- 변경 관리 / 교육

도메인 분석
- 이벤트 스토밍 도구 활용
- 비즈니스 도메인 분석
- 도메인 별 호출 관계 분석

서비스 식별

바운디드 컨텍스트 분류
- 분석된 도메인별 분류

애그리게잇
- 커맨드 식별
- 이벤트 식별
- 모델링

설계 구축

애플리케이션 설계
- 개발 프레임워크 선정 (폴리글랏 또는 표준)
- Backing 서비스 선정
- 내부 통신 방식 설계
- DB 분리 및 SAGA 패턴 설계

인프라 설계 및 구축
- CSP 및 서비스 선택
- 보안 솔루션 선정 및 구축
- 서비스 메시
- 로그 수집 및 분석 환경 구축
- 배포 파이프라인 설계 및 구축

서비스 개발

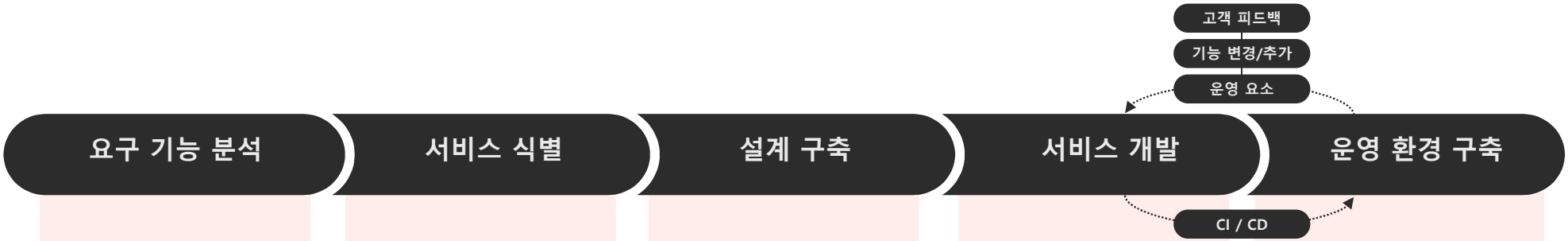
애플리케이션 개발
- 도메인 별 비즈니스 로직 개발
- 기존 시스템간 호환 로직 처리
- 서비스 간 트랜잭션 처리
- K8s 배포 리소스 개발 (서비스 의존성 컨테이너 주입)
- 서비스 로깅 체계 개발

운영 환경 구축

모니터링
- VM 단위 리소스
- POD 단위 리소스
- 서비스 단위 메트릭 선정
- 각 메트릭 별 임계치 설정
- 알림 체계 구축

관찰 가능성
- 복잡한 시스템에 대한 가시성 확보
- 잠재된 위험 요소 실시간 감지
- 서비스 간 호출 흐름(종속성) 분석

How to modernization? 기존 시스템 전환(마이그레이션)은 어떻게 하나요?



AS-IS 분석
- 서비스 개발 환경
- 서비스 실행 환경
- 서비스 운영 환경

요구 사항 분석
(신규 시스템 개발 동일)

도메인 분석
- 이벤트 스토밍 도구 활용
- 비즈니스 도메인 분석
- 도메인별 호출 관계 분석

바운디드 컨텍스트 분류
- 분석된 도메인별 분류

애그리게잇
- 커맨드 식별
- 이벤트 식별
- 모델링

애플리케이션 설계 — 기존 서비스 개발 환경
- 개발 프레임워크 선정 (폴리글랏 또는 표준)
- Backing 서비스 선정
- 서비스/외부시스템 통신 방식
- DB 분리 / 동기화 전략 설계
- DB 마이그레이션 (이기종 간)
- DB 분리에 따른 SAGA 패턴 설계

인프라 설계 및 구축 — 기존 서비스 개발 환경
- CSP 및 서비스 선택
- 보안 솔루션 선정 및 구축
- 서비스 메시
- 로그 수집 및 분석 환경 구축
- 배포 파이프라인 설계 및 구축

애플리케이션 개발
- 도메인별 비즈니스 로직 개발
- 기존 시스템간 호환 로직 처리
- 서비스간 트랜잭션 처리
- K8s 배포 리소스 개발 (서비스 의존성 컨테이너 주입)
- 서비스 로깅 체계 개발

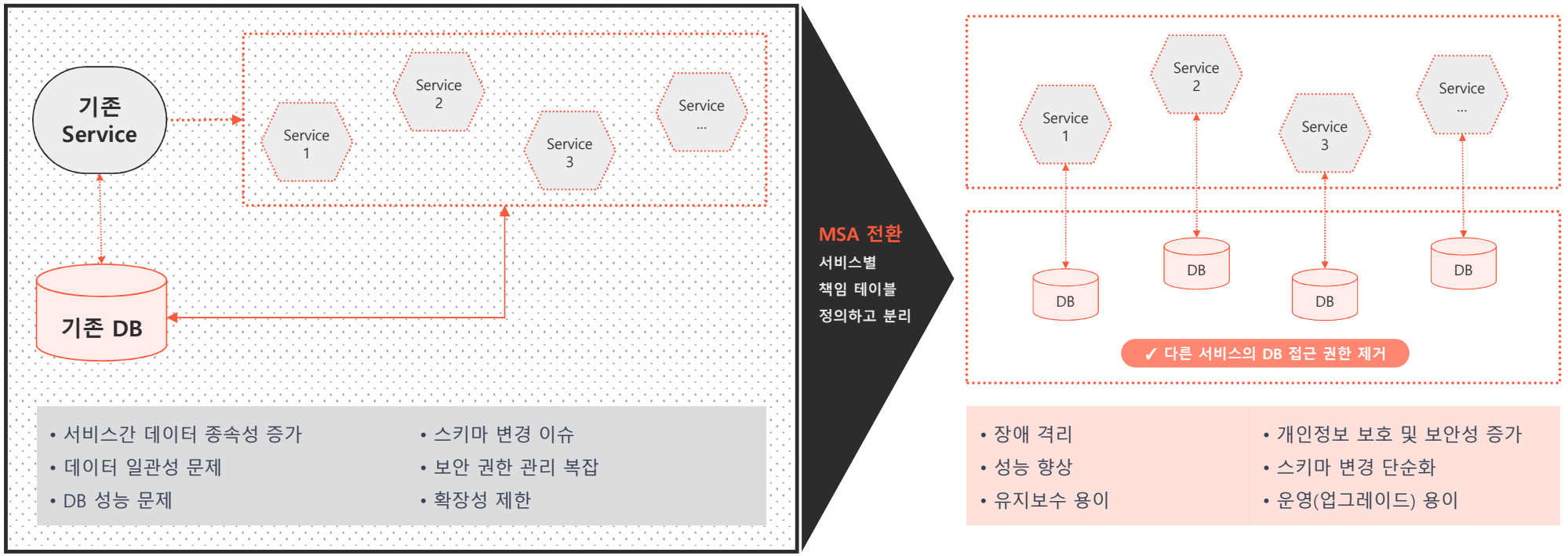
모니터링 — 기존 서비스 운영 환경
- VM 단위 리소스
- POD 단위 리소스
- 서비스 단위 메트릭 선정
- 각 메트릭 별 임계치 설정
- 알림 체계 구축

관찰 가능성
- 복잡한 시스템에 대한 가시성 확보
- 잠재된 위험 요소 실시간 감지
- 서비스간 호출 흐름(종속성) 분석

How to modernization? 데이터베이스 전환은 어떻게 하는게 좋을까요? I

MSA로 전환하려면 데이터베이스 분리는 필수

- 상용DB에서 오픈소스 DB로 전환 (DB 마이그레이션)
- 서비스별 DB간 데이터 동기화 전략 - 기존 로직 및 PL/SQL에서 테이블간 종속성 해결
  - 다른 서비스에 데이터 요청 후 처리하는 코드 또는 쿼리 재사용



How to modernization? 데이터베이스 전환은 어떻게 하는게 좋을까요? II

- MSA로 전환하려면 데이터베이스 분리는 필수
- **상용DB에서 오픈소스 DB로 전환 (DB 마이그레이션)**
- 서비스별 DB간 데이터 동기화 전략 - 기존 로직 및 PL/SQL에서 테이블간 종속성 해결
  - 다른 서비스에 데이터 요청 후 처리하는 코드 또는 쿼리 재사용

기존 DB 분석	대상 DB 선택	데이터 준비	마이그레이션	로직/쿼리 마이그레이션
AS-IS 분석 • 기존 스키마 • 데이터 타입 • 쿼리 / 프로시저 / 함수	**오픈소스 DB 선정** • 비용, 기능 효율적 여부 • 호환성, 성능, 지원 여부	**데이터 정규화 및 분류** • 기존 DB 백업 • 적재성, 배치성, 트랜잭션용 데이터 분류	**스키마/데이터 마이그레이션** • 호환 데이터 타입 변경 • 스키마 생성 및 구성 • 다운타임 고려 (일괄, 점진적) • 데이터 정합성 검증	**로직 수정 및 쿼리 검토** • 대상 DB 쿼리로 최적화 • PL/SQL 변경 • 애플리케이션 로직 수정 • 쿼리 최적화 : 실행계획 분석 : 인덱스, 튜닝 등

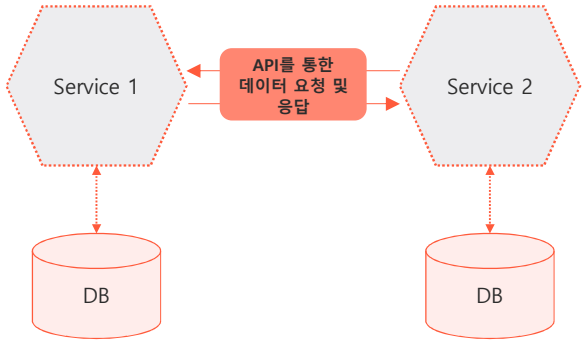
※ MSA에서는 상용 DB 사용시 솔루션 비용이 증가 → 오픈소스 DB로의 전환이 필수적

How to modernization? 데이터베이스 전환은 어떻게 하는게 좋을까요? III

- MSA로 전환하려면 데이터베이스 분리는 필수
- 상용DB에서 오픈소스 DB로 전환 (DB 마이그레이션)

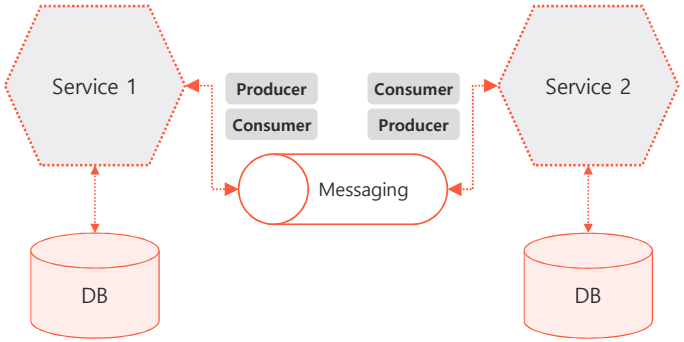
서비스별 DB간 데이터 동기화 전략 - 기존 로직 및 PL/SQL에서 테이블간 종속성 해결
- 다른 서비스에 데이터 요청 후 처리하는 코드 또는 쿼리 재사용

API 방식 (동기식)



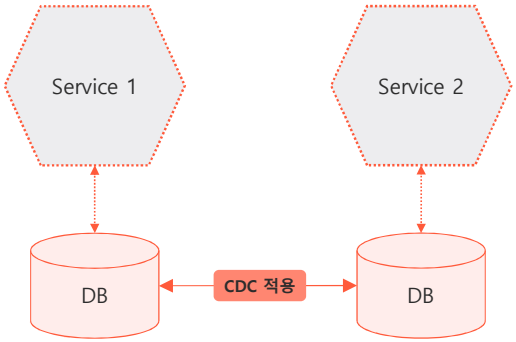
- 데이터 처리 로직 복잡
- 스키마 변경에 따른 로직 변경
- API 호출로 인한 Latency 증가
- 기존 작성된 쿼리나 코드의 재사용성 저하

Messaging 채널 방식 (비동기식)



- 각 서비스별 공통 메시징 처리 로직 추가
- 비동기 처리 로직이 많은 경우 적합
- 인프라 추가 및 모니터링 등 운영 복잡도 증가

CDC (Change Data Capture) 방식



- 준 실시간 데이터 변경
- 증분 변경 사항 감지 적용
- DB 부하 적음

How to modernization? 데이터베이스 전환은 어떻게 하는게 좋을까요? III

API 방식 (동기식)

복잡하거나 특정한 쿼리의 결과

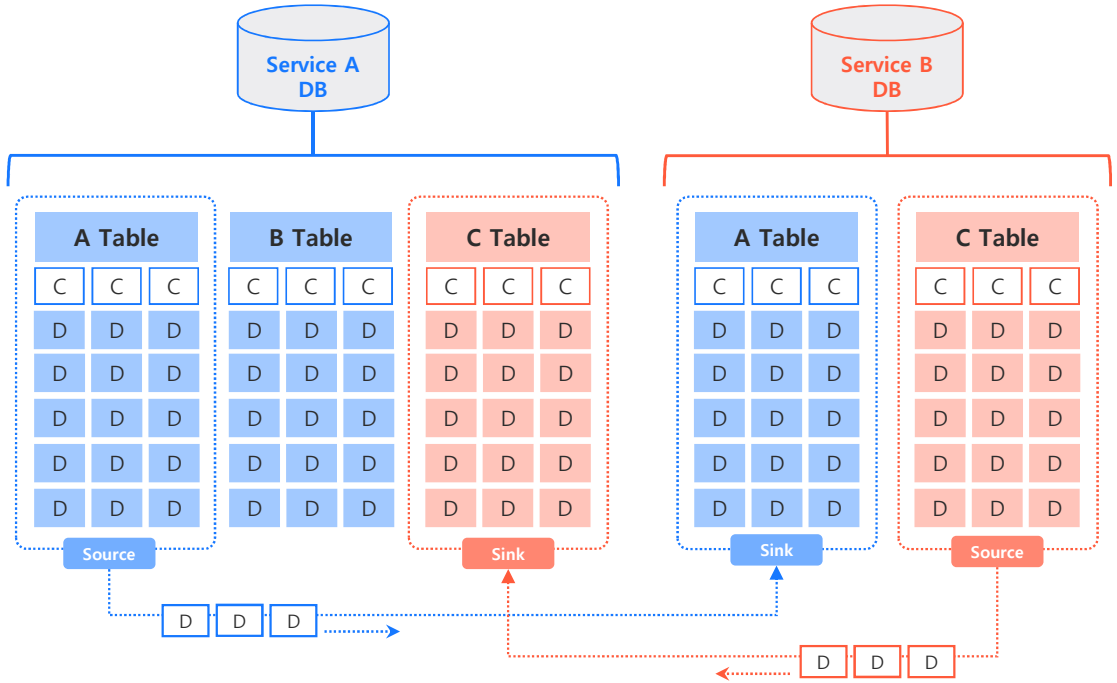
비교적 간단한 데이터 조회 작업일 경우

여러 테이블과의 Join등이 필요 없는 데이터일 경우

기타 대상 서비스에서만 획득할 수 있는 데이터일 경우

+

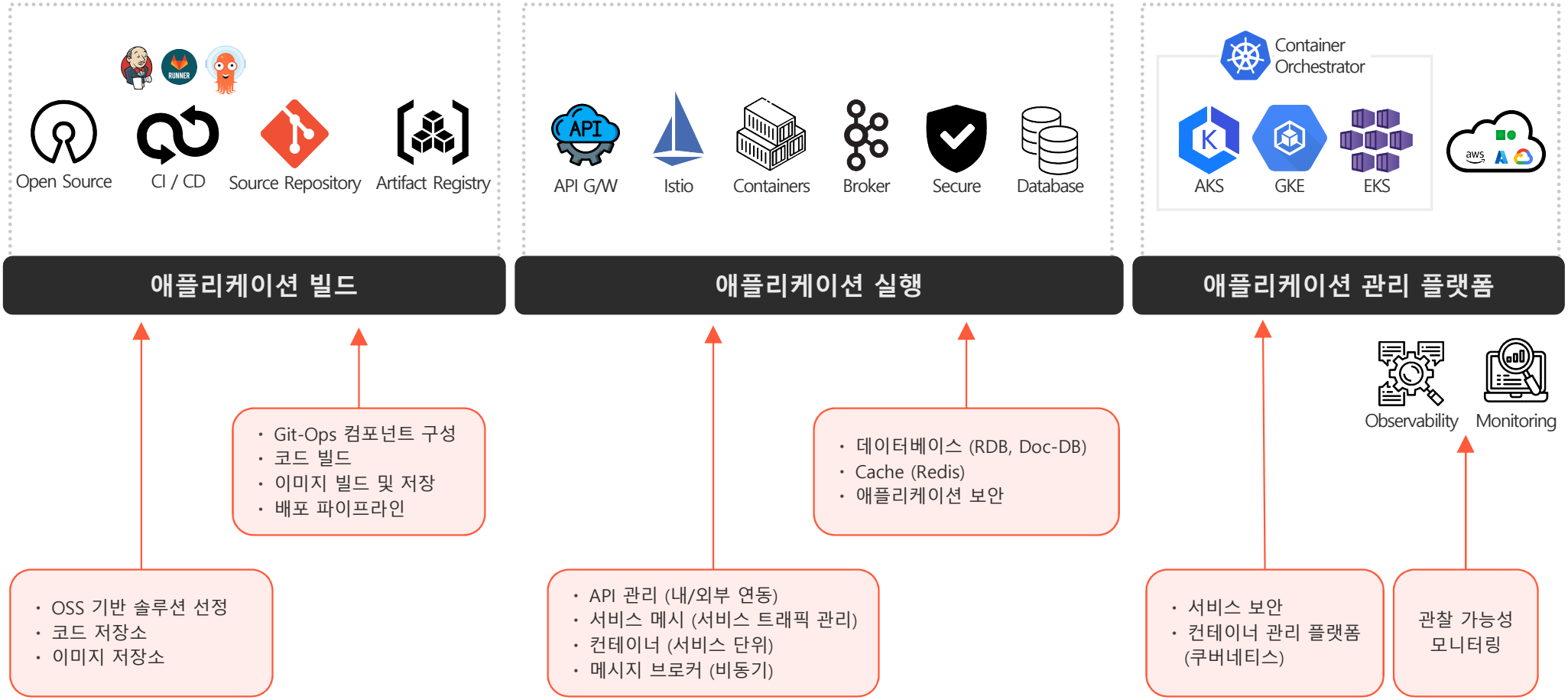
CDC (Change Data Capture) 방식



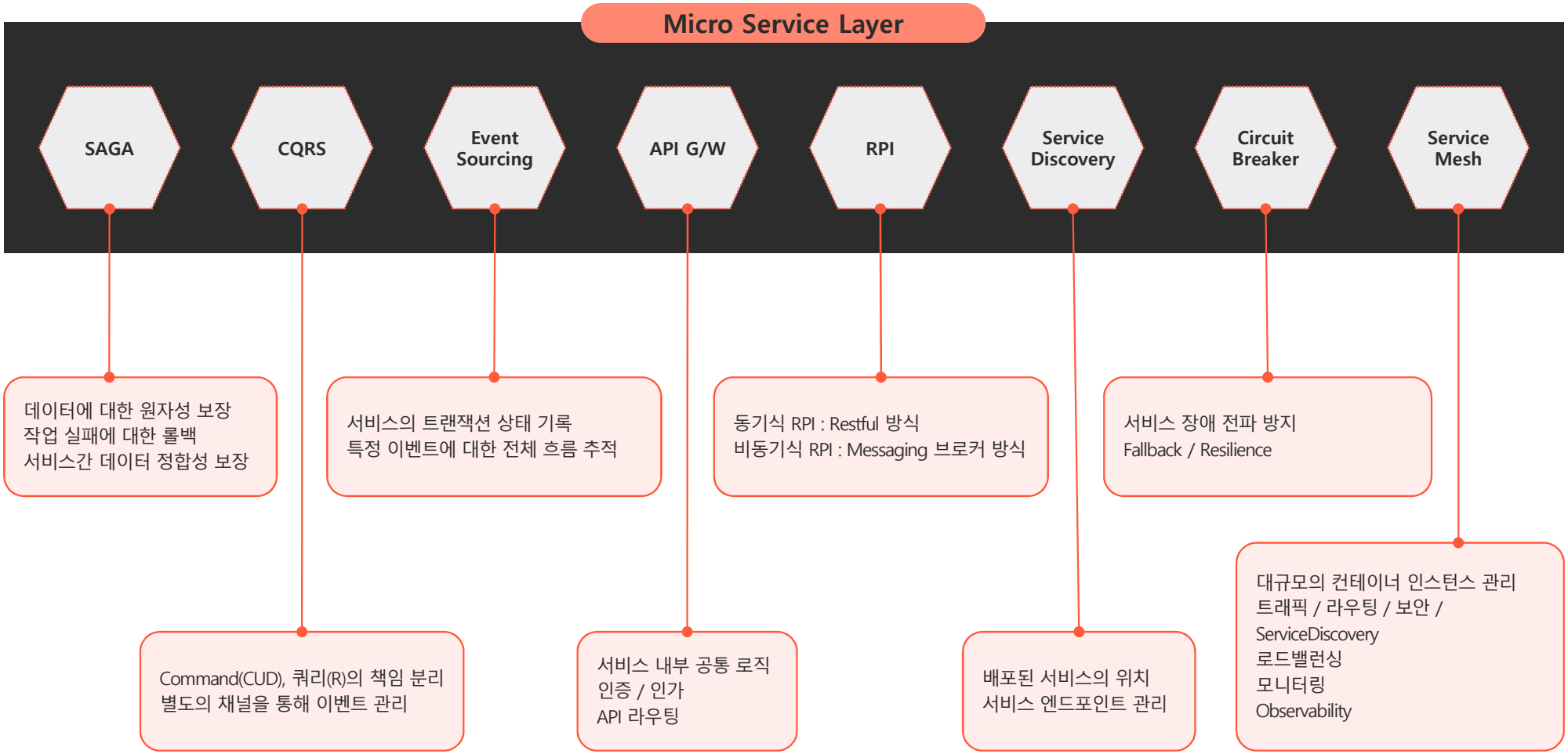
04 MSA를 위한 필수 요소

- 컴포넌트
- 기술(Technology)

MSA를 위한 컴포넌트 MSA 구축에 사용되는 컴포넌트는 어떤 것들이 있나요?



기술(Technology) MSA에는 어떤 기술들이 필요한가요?

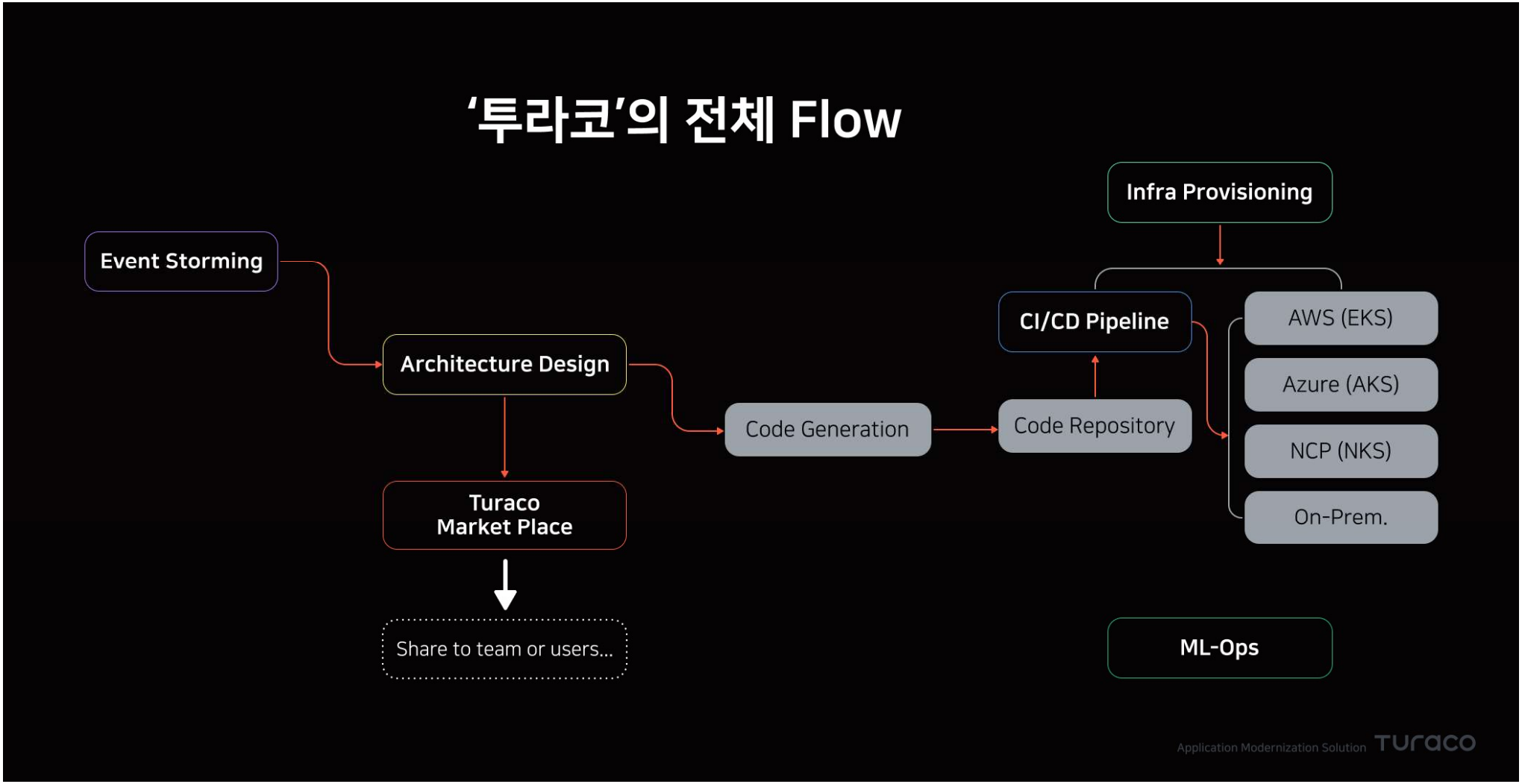


05 투라코(Turaco) 솔루션 소개

- 투라코(Turaco)의 핵심
- 전체 Flow
- 주요기능

투라코(Turaco) 투라코를 소개합니다.

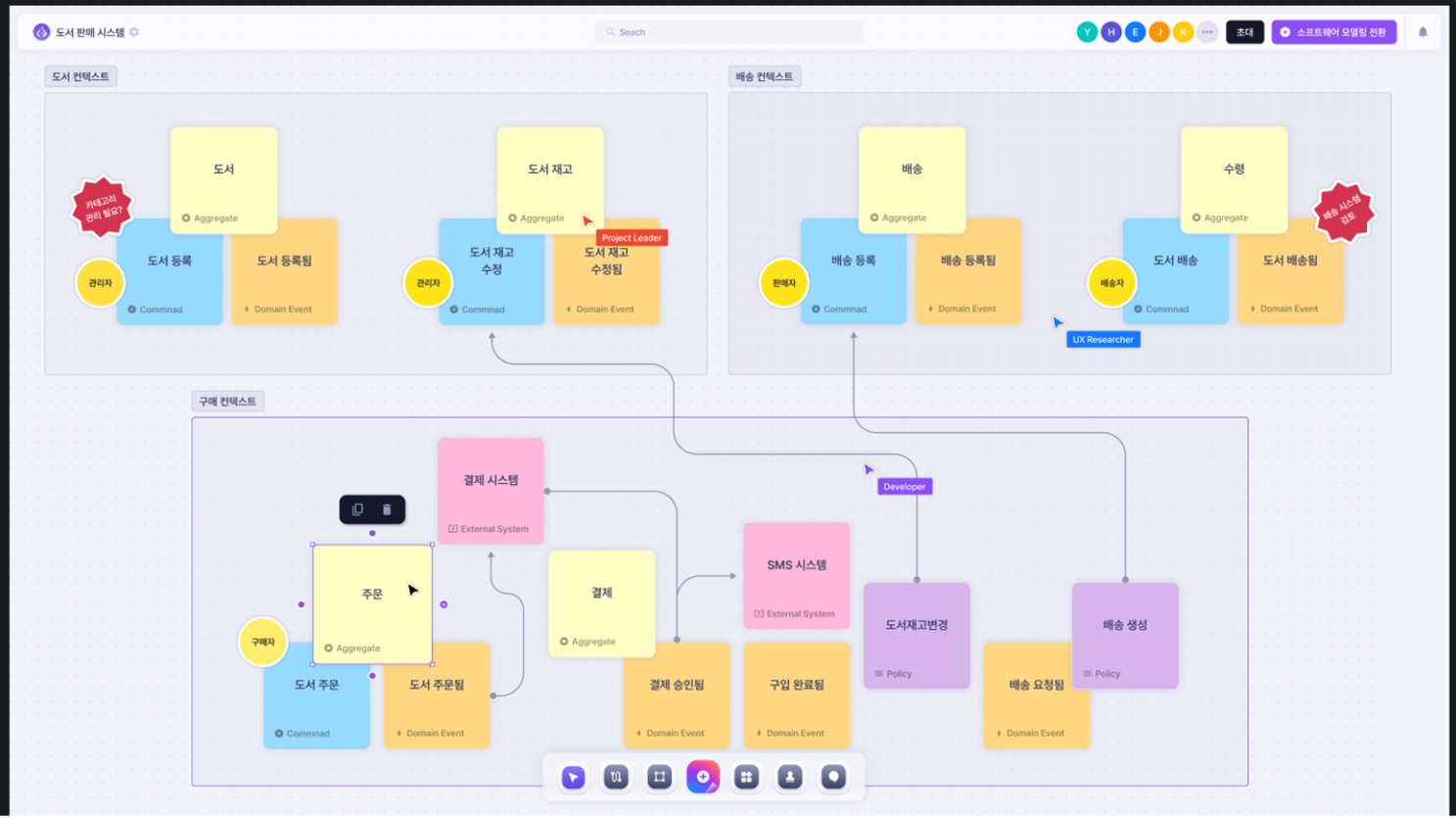
투라코(Turaco) 도메인 분석부터 배포 모니터링까지...



투라코(Turaco) 투라코를 소개합니다.

투라코(Turaco) 이벤트 스토밍

DDD를 위한 전략적 설계 도출



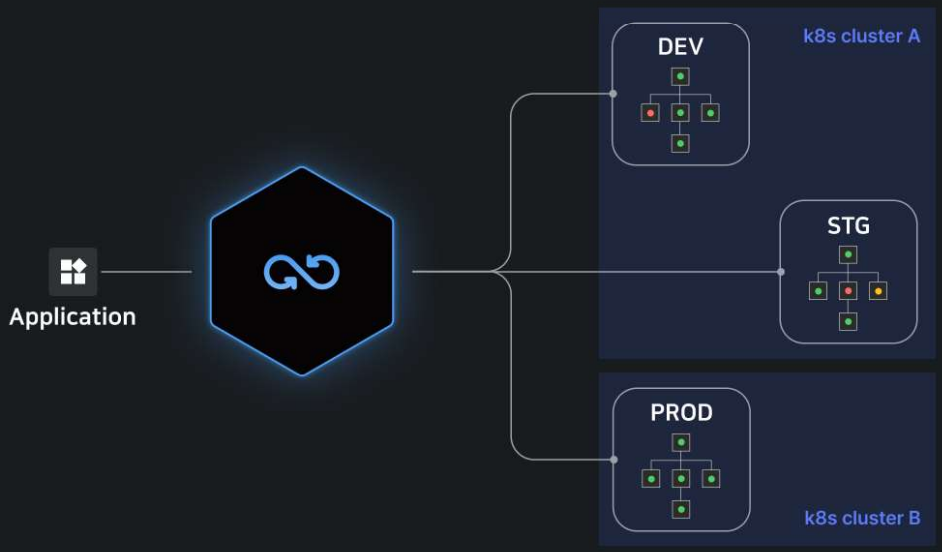
투라코(Turaco) 아키텍처 디자이너

투라코(Turaco) DevOps & CI/CD 파이프라인

빠른 DevOps 환경 구축

DevOps 자동화
소스 코드 저장소부터 빌드, 배포 및
모니터링 전 과정을 쉽고 빠르게 구축

다중 환경
하나의 애플리케이션으로 여러 클러스터
또는 스테이지에 배포 가능



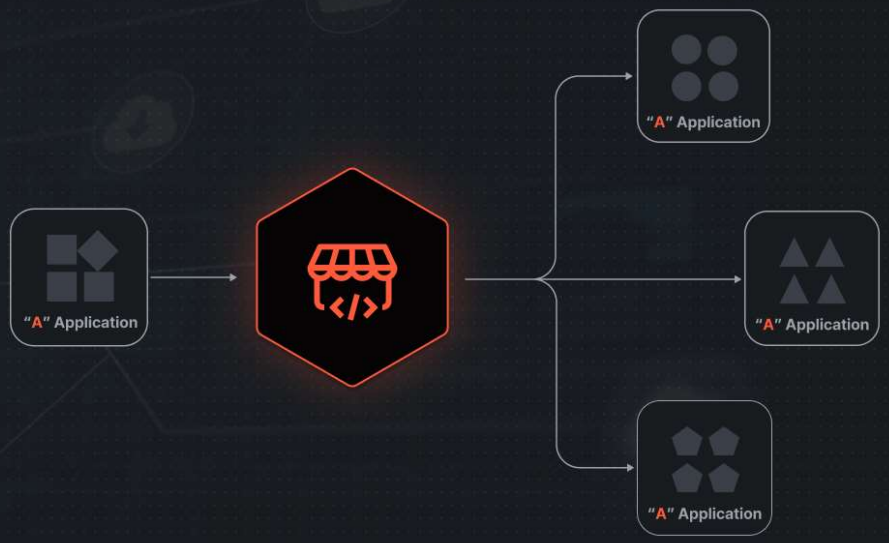
Application Modernization Solution Turaco

투라코(Turaco) 마켓 플레이스

마켓 플레이스를 통한 소스 공유 및 재사용

마켓플레이스
완성된 개발 애플리케이션과
구성한 아키텍처 공유 및 재사용

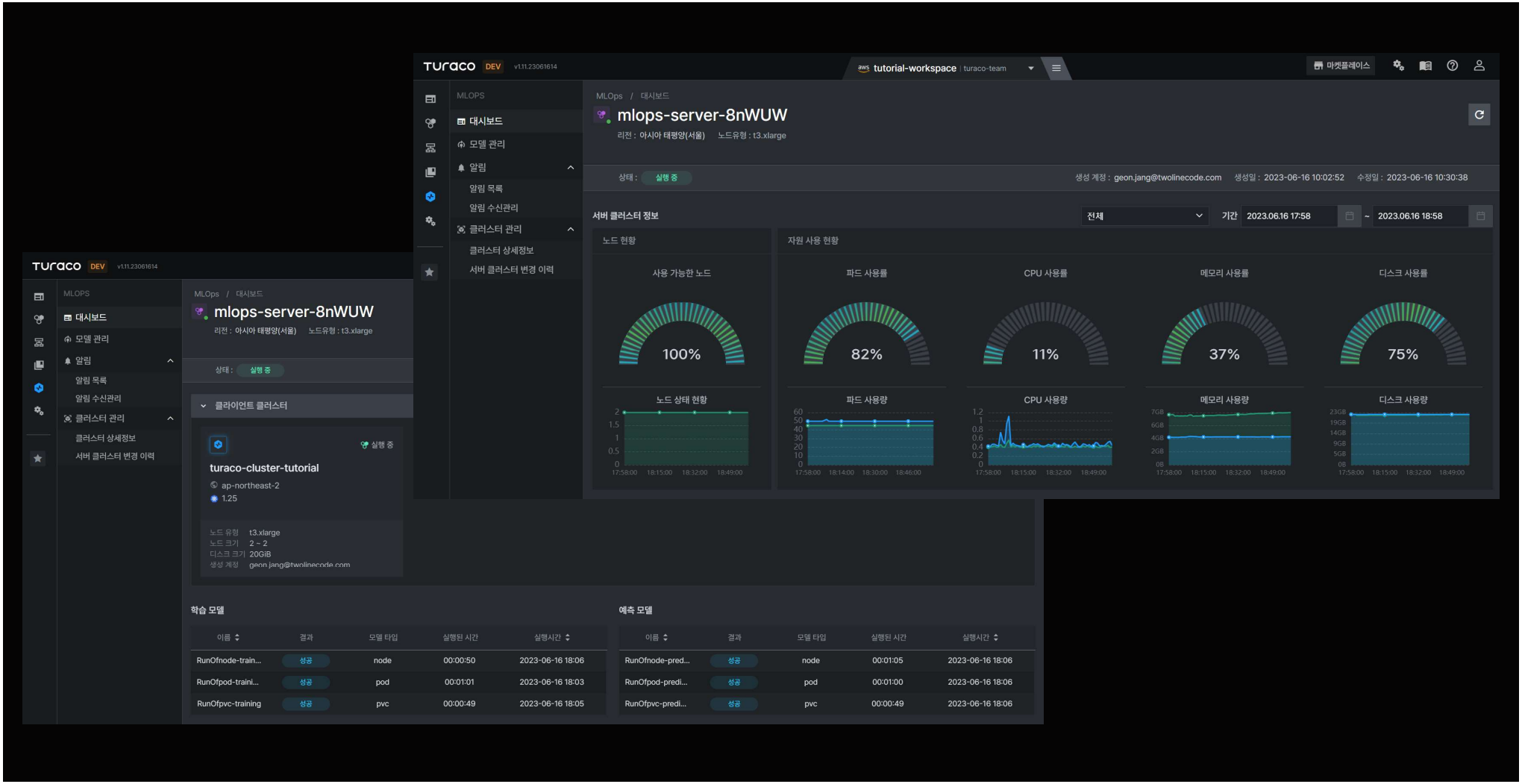
사업별 템플릿
산업별로 제공된 서비스 템플릿
활용으로 개발기간 단축



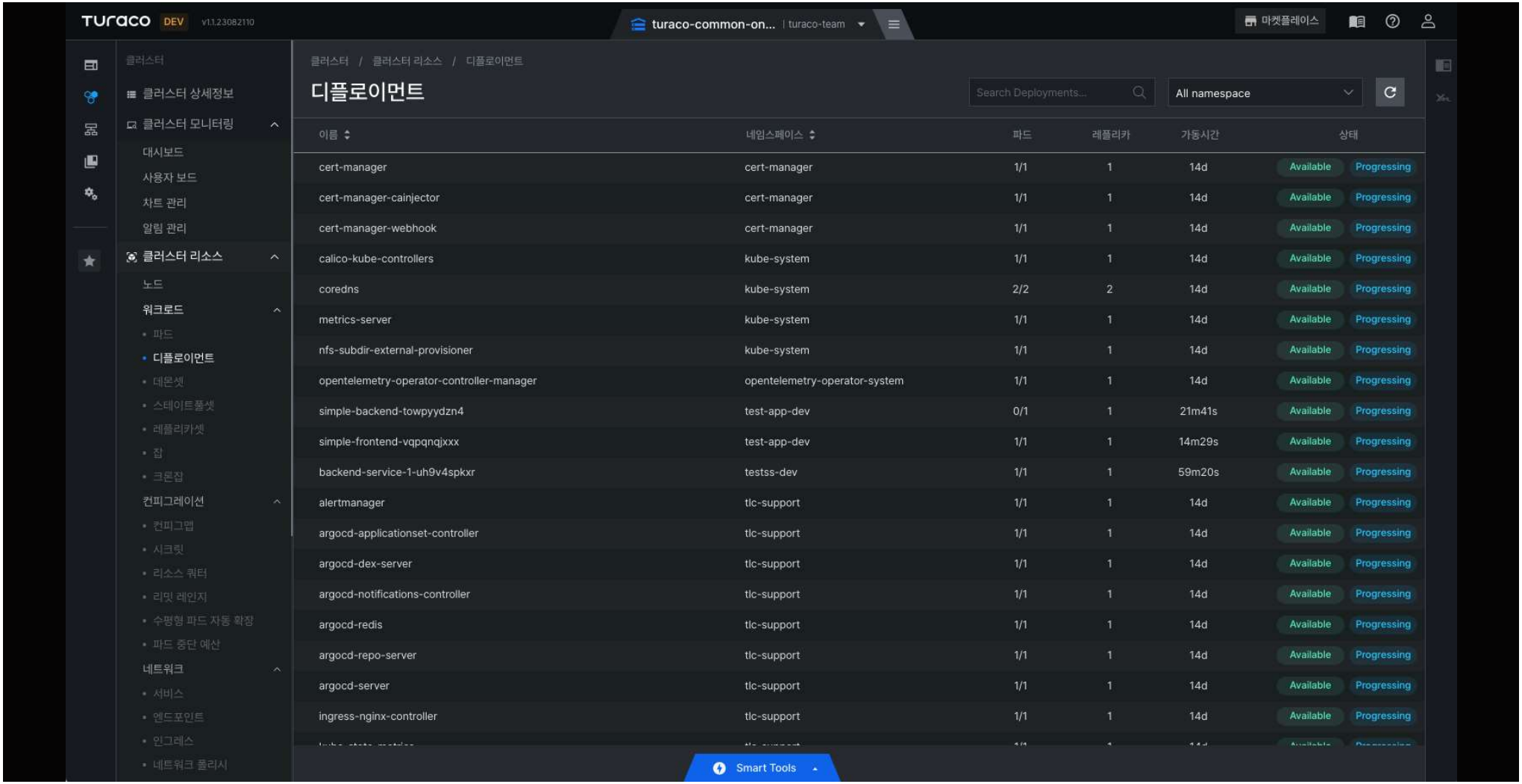
Application Modernization Solution turaco

투라코(Turaco) MLOps

투라코(Turaco) 클러스터 모니터링



투라코(Turaco) 클러스터 리소스 관리



Perfect Support, Turaco

투라코는 Application Modernization 전환의 필수 요건들을 지원하여
쉽고 빠르게 비즈니스 요구사항을 대응합니다.

www.turacocloud.com

Twolinecode

투라인코드는 퍼블릭 클라우드 (AWS, GCP, Azure) 기반 서비스 / 플랫폼 개발 및 운영 등에 대한 다양한 지식과 오랜 경험을 바탕으로 성장해 왔으며 빅데이터 수준의 분석 솔루션 및 컨테이너 기반의 마이크로 서비스 개발 그리고 언택트 통합솔루션을 제공하는 IT 전문 기업입니다.

- 투라인코드는 Cloud 시장에서의 독보적인 전문가 집단을 목표로 2013년에 설립
- 설립 이후 AWS와 Google등 전문 Cloud Service Provider의 컨설팅 파트너 자격을 획득하는 등 2016년까지 Cloud 사업 수행을 위한 기반을 확보
- 삼성, LG, SK 등 국내 Top 대기업 주요 계열사의 Cloud 기반 시스템 구축 사업의 파트너로써 수많은 경험을 바탕으로 Cloud Native 시장에서 인정을 받았음
- 2019년 이후 SK의 Cloud 및 AI 영역의 플랫폼 개발 사업에 참여함으로써 플랫폼 구현의 역량을 확보하였고, 그 경험을 바탕으로 독자적인 Turaco를 개발
- Turaco 뿐만 아니라, VDI 등 Digital 기반 다양한 플랫폼 사업으로 확장을 추진중

회사명	(주)투라인코드 (Twolinecode Inc.)
대표이사	현승엽
설립일	2013년 3월
Head Office	경기도 성남시 분당구 성남대로 331번길 11-3, (정자동) 여민빌딩 5층
R&D Center	경기도 성남시 분당구 성남대로 343번길 12-2, 신수빌딩 3층
R&D Center	서울시 강남구 테헤란로 2길 27 패스트파이브 강남 5호점 13층
New York Office	127 West 30th St. Unit #1014 New York, NY 10001 USA

Thank you