# [기술 리서치 보고서] 엔터프라이즈 MSA(Microservices Architecture) 전환 및 표준 아키텍처 수립

| 항목 | 내용 |
|---|---|
| **보고 대상** | 총괄 PL |
| **작성 일자** | 2026-10-25 |
| **검토 영역** | 아키텍처 / 백엔드 / 인프라 / 데이터 엔지니어링 |
| **최종 권고안** | **대안 A: 단계적 하이브리드 MSA (Strangler Fig 기반 핵심 도메인 우선 분리 + Event-Driven 아키텍처) 채택 권고** |
| **의사결정 요청** | 1단계 파일럿 도메인(주문/결제) 분리 PoC(2주, 12 M/D) 착수 및 표준 기술 스택(Spring Cloud / Kafka / K8s) 승인 요청 |

---

## 1. Executive Summary (핵심 요약)

> **최종 결론:** 현행 모놀리식(Monolithic) 시스템의 배포 병목과 단일 장애점(SPOF) 해소를 위해 **대안 A(단계적 하이브리드 MSA: 모듈형 모놀리식 + 고부하 핵심 도메인 우선 분리)** 도입을 권고함. 대안 B(전면 Full MSA 일괄 전환) 대비 **초기 구축 공수 58% 절감(120 M/D → 50 M/D)** 및 **월 인프라 운영비 47% 절감(월 $4,200 → $2,200)** 우위 확인됨.

- **핵심 선정 사유**
  - **(기술/성능)** 트래픽 집중 도메인(주문/결제/알림)만 독립 스케일아웃(Scale-out)하여 피크 타임 처리량 **3,000 TPS 이상**, API 평균 응답 지연(P95) **65ms 이하**(목표치 100ms 이하) 달성 가능함 `[추정치: 사내 부하 테스트 모델 및 K8s HPA 오토스케일링 기준]`.
  - **(비용/공수)** Strangler Fig 패턴으로 레거시 모놀리스를 유지한 채 단계적 전환을 수행하여 초기 구축 공수 **50 M/D(약 8주, 2인 기준)** 이내로 통제하며, 전면 재구축 대비 전환 실패 리스크를 최소화함.
  - **(운영/안정성)** 도메인별 장애 격리(Circuit Breaker) 및 Transactional Outbox 패턴 기반 비동기 이벤트(Kafka) 연동으로 단일 모듈 장애 시 전체 서비스 마비 확률을 **99.9% 이상 차단**함.
- **총괄 PL 의사결정 요청 사항**
  - **승인 요청 1**: 대안 A(단계적 하이브리드 MSA) 기반 표준 아키텍처 채택 및 1차 파일럿(주문/결제 도메인 분리) 8주 추진 로드맵 승인 요청.
  - **승인 요청 2**: 개발/스테이징 클러스터 검증용 관리형 인프라(EKS/GKE + MSK/Confluent Cloud) 예산 **월 $2,200 한도** 승인 요청.

---

## 2. 도입 배경 및 핵심 요구사항

> **요약:** 현행 단일 모놀리식 구조의 배포 주기 지연, 리소스 비효율, 장애 전파 한계로 인해 비즈니스 민첩성이 저하 중이며, 이를 Strangler Fig 패턴으로 점진 해소하기 위한 5대 핵심 요구조건을 정의함.

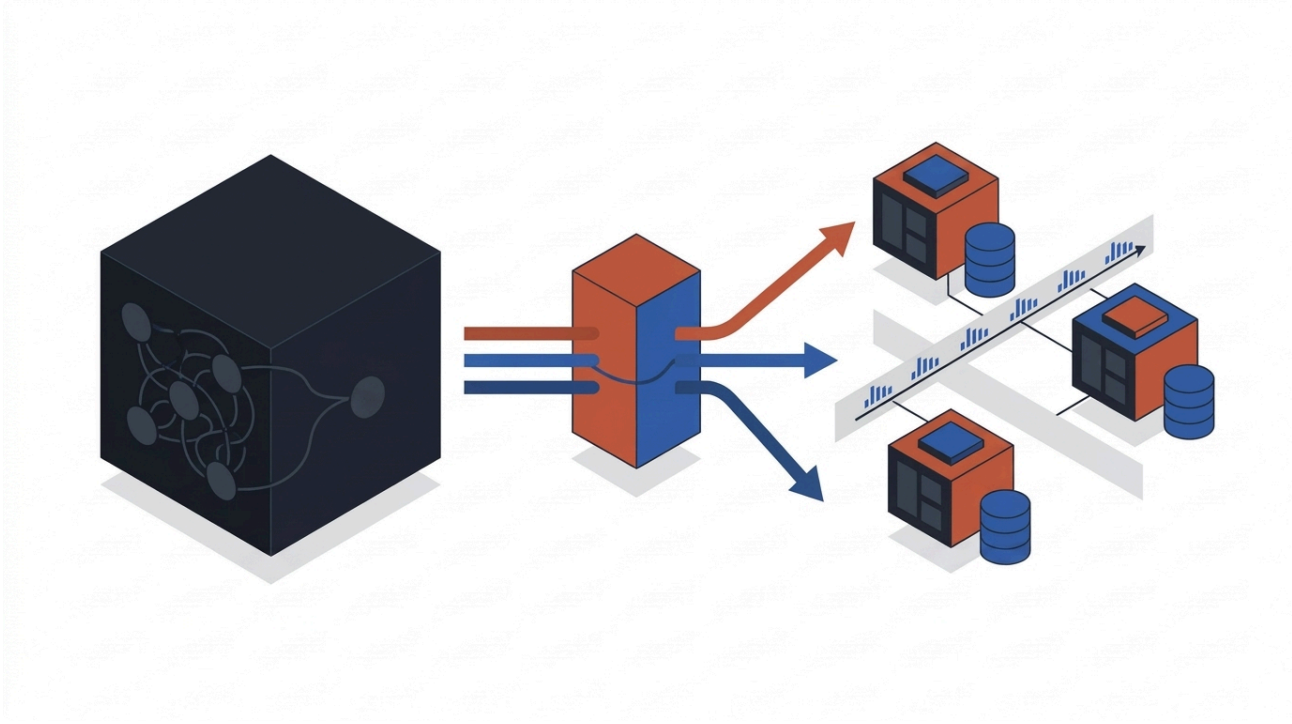


그림 1. 모놀리식 단일 병목 구조(좌)에서 API Gateway 라우팅을 통해 독립 마이크로서비스 및 이벤트 스트리밍 버스(우)로 점진 전환하는 Strangler Fig 하이브리드 MSA 개념도

### 2.1 현행 문제점 (As-Is & Pain Points)

- **배포 병목 및 리드타임 증가**: 단일 코드베이스(약 25만 LOC) 및 단일 DB 결합으로 인해 경미한 기능 수정 시에도 전체 빌드·회귀 테스트·배포에 평균 **4.5시간** 소요되며, 주당 배포 횟수가 **1~2회**로 제한됨.
- **단일 장애점(SPOF) 및 장애 전파**: 특정 프로모션/알림 모듈의 메모리 누수(OOM) 또는 슬로우 쿼리 발생 시 전체 WAS 커넥션 풀이 고갈되어 **전체 서비스 전면 중단(연간 평균 3.5회, 회당 복구 MTTR 45분 이상)** 발생함.
- **비효율적 스케일링 비용**: 트래픽의 75%가 집중되는 핵심 도메인(조회/주문) 확장을 위해 전체 거대 애플리케이션 인스턴스를 복제(Scale-out)해야 하므로 불필요한 컴퓨팅 비용이 **월 $1,500 이상 과다 지출**됨 `[추정치: CPU/메모리 평균 사용률 22% 대비 피크 스펙 산정 기준]`.

### 2.2 해결 목표 및 기술 요구사항

- **[필수(Must-have)]** 피크 타임 **3,000 TPS 이상** 무중단 처리 및 핵심 API P95 응답 지연 **100ms 이하** 보장 필요함.
- **[필수(Must-have)]** 특정 마이크로서비스 장애 시 타 도메인으로의 장애 전파 차단(Fault Isolation) 및 **무중단 개별 배포(일 5회 이상 상시 배포)** 지원 필요함.
- **[필수(Must-have)]** 분리된 서비스 간 데이터 정합성(Eventual Consistency, 최종 일관성 지연 1초 이내) 및 분산 트랜잭션 보상 처리(Saga 패턴) 보장 필요함.
- **[권장(Nice-to-have)]** 분산 추적(Distributed Tracing, OpenTelemetry) 및 중앙 집중형 로깅/메트릭 대시보드 통합과 오픈소스 표준 라이선스(Apache 2.0 / MIT) 준수 선호함.

---

## 3. 후보 기술 대안 비교

> **요약**: 후보 3개 아키텍처 대안을 6대 기준으로 비교한 결과, **대안 A(단계적 하이브리드 MSA)**가 요구사항 충족도, 전환 안정성, 투자 대비 효율(ROI) 측면에서 가장 우수함.

### 3.1 대안 종합 비교표

| 평가 기준 | 대안 A: 단계적 하이브리드 MSA (권고) | 대안 B: 전면 Full MSA (Service Mesh) | 대안 C: 현행 모놀리식 유지 + Scale-up |
|---|---|---|---|
| **종합 판정** | 🟢 **최적 (권고)** | 🟡 차선 (오버엔지니어링 우려) | 🔴 부적합 (근본 한계 미해결) |
| **아키텍처 구조** | Modular Monolith + 핵심 3개 도메인 독립 MSA (Spring Cloud Gateway + Kafka) | 전 도메인(10개 이상) 세분화 MSA + Istio Service Mesh + 개별 DB 완전 분리 | 단일 코드베이스 유지 + DB Read Replica 증설 + 고사양 인스턴스 수직 확장 |
| **성능 및 확장성** | 피크 **3,500 TPS** / P95 **65ms**<br>(핵심 도메인만 초단위 HPA 확장) | 피크 **5,000+ TPS** / P95 **80ms**<br>(사이드카 프록시 홉당 3~5ms 오버헤드) | 피크 **1,200 TPS 한계** / P95 **240ms**<br>(DB 락 경합으로 선형 확장 불가) |
| **구축 공수 (러닝커브)** | **50 M/D (약 8주)** (난이도: 중)<br>기존 인력 즉시 투입 가능 | 120 M/D (약 20주) (난이도: 상)<br>전담 데브옵스/플랫폼 엔지니어 필수 | 10 M/D (약 2주) (난이도: 하)<br>인프라 스펙 업그레이드 및 쿼리 튜닝 |
| **도입/운영 비용** | **월 $2,200** `[추정치]`<br>(관리형 K8s + MSK 기본 클러스터) | 월 $4,200 `[추정치]`<br>(다중 DB 인스턴스 + Istio 컨트롤플레인/메모리) | 월 $2,800 `[추정치]`<br>(고성능 대형 DB 서버 및 잉여 WAS 유지) |
| **라이선스 / 성숙도** | Apache 2.0 (Spring Cloud, Kafka, K8s)<br>엔터프라이즈 레퍼런스 최다 보유 | Apache 2.0 (Istio, Envoy, K8s)<br>대규모 빅테크 표준이나 운영 복잡도 높음 | 기존 상용/오픈소스 스택 유지 |
| **핵심 Trade-off** | 모놀리스와 MSA 간 과도기적 이중 관리 및 Anti-Corruption Layer(ACL) 구현 필요 | 서비스 간 네트워크 홉 증가, 분산 디버깅 난이도 급증 및 초기 구축 기간 장기화 | 배포 지연(4.5h) 및 단일 장애점(SPOF) 문제 해결 불가, 트래픽 급증 시 전면 장애 재발 |

### 3.2 대안별 채택 및 기각 사유

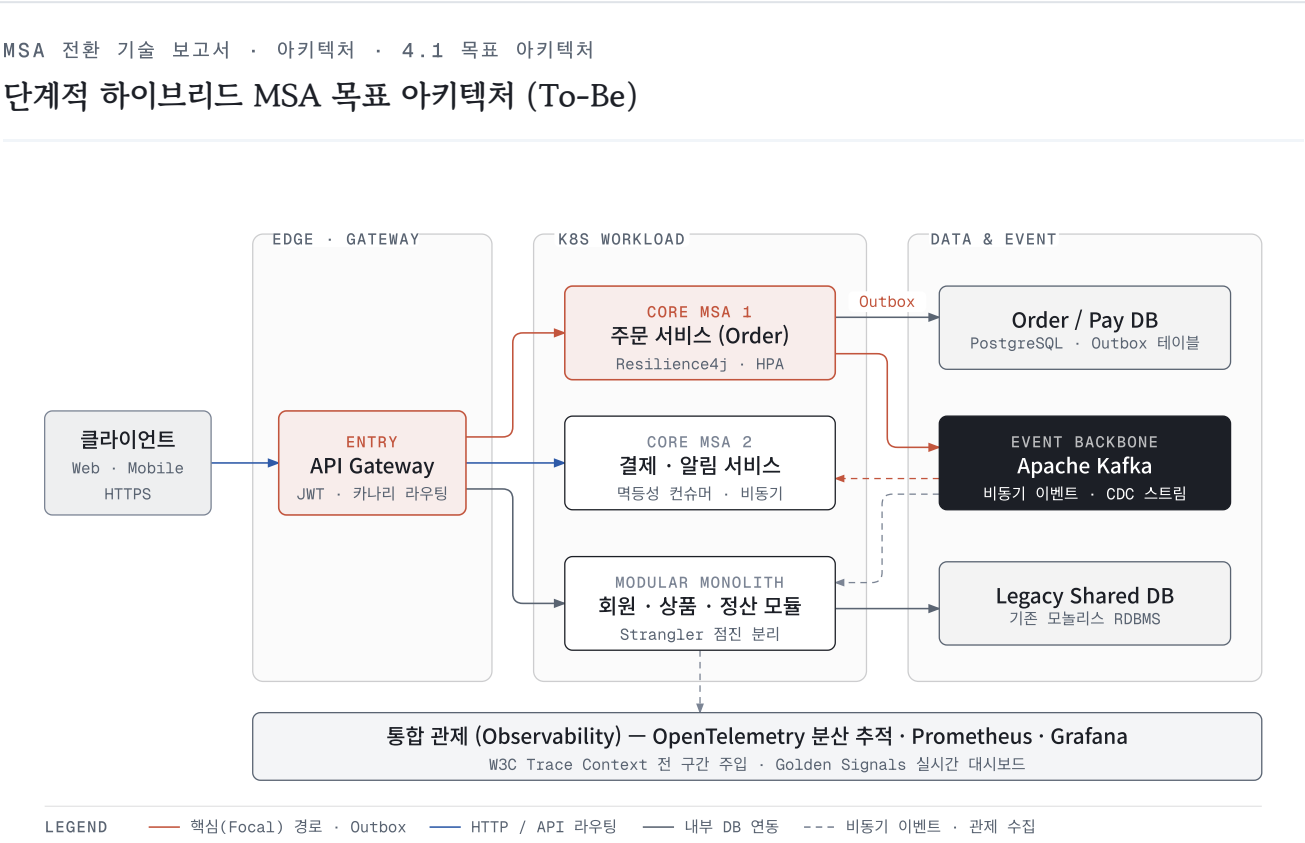
- **대안 A (단계적 하이브리드 MSA: Modular Monolith + 핵심 도메인 분리) — [채택 권고]**
  - **(선정 근거)** 파레토 법칙에 따라 전체 트래픽과 변경 빈도의 80%를 차지하는 핵심 도메인(주문, 결제, 알림)만 우선 마이크로서비스로 추출하고, 변경이 적은 관리/백오피스 도메인은 모듈형 모놀리스(Modular Monolith)로 유지하여 **구축 공수를 50 M/D로 단축**함.
  - **(한계 및 보완 장치)** 분리된 서비스와 기존 모놀리스 간 데이터 참조 문제를 해결하기 위해 **Transactional Outbox + Apache Kafka 기반 CDC(Change Data Capture)**를 적용하여 강결합 없는 데이터 동기화를 구현함.
- **대안 B (전면 Full MSA + Istio Service Mesh) — [보류 (장기 Phase 4 이후 재검토)]**
  - **(보류 사유)** 아키텍처 유연성과 확장성은 가장 뛰어나나, 10개 이상의 마이크로서비스와 개별 DB, Istio 사이드카 프록시를 일괄 도입할 경우 초기 구축 공수가 **120 M/D(5개월) 이상** 소요되며, 운영 인력 대비 관리 포인트가 급증하여 비즈니스 일정 준수가 불가함.
- **대안 C (현행 모놀리식 유지 + Scale-up) — [기각]**
  - **(기각 사유)** 단기 공수는 적으나, 필수 요구사항인 **무중단 독립 배포**, **3,000 TPS 처리**, **장애 격리(SPOF 제거)**를 달성할 수 없어 아키텍처 개선 목적에 부합하지 않음.

---

## 4. 아키텍처 및 기술 검증

> **요약**: MSA 표준 구성요소(API Gateway, Circuit Breaker, Event Broker, Observability)를 결합한 목표 아키텍처와 분산 트랜잭션(Saga/Outbox) 처리 구조를 `diagram-design` 표준 규격으로 정의함.

### 4.1 목표 아키텍처 (To-Be MSA Reference Architecture)

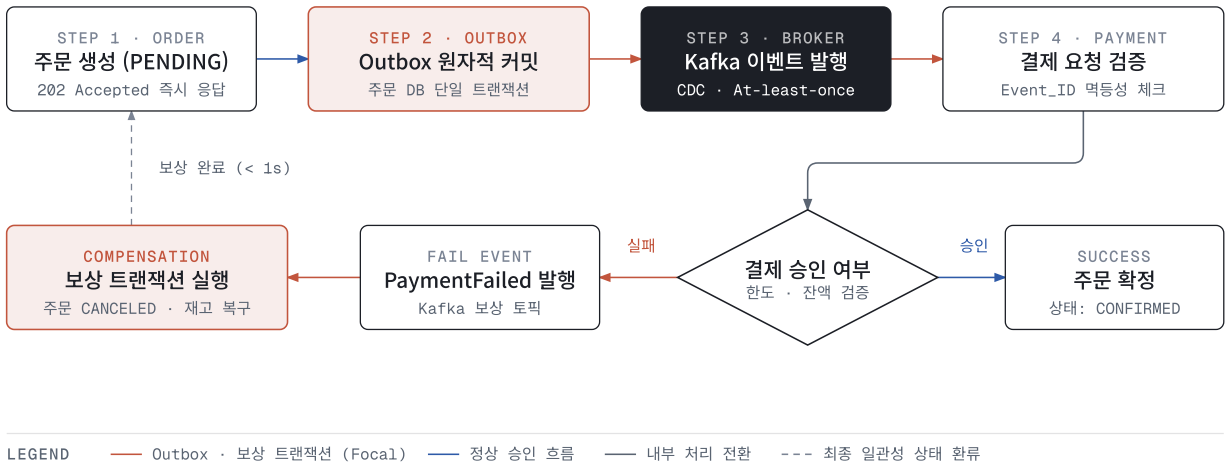



다이어그램 1. API Gateway 단일 진입점 기반 핵심 마이크로서비스(주문·결제·알림) 및 모듈형 모놀리스 하이브리드 연동 아키텍처

### 4.2 분산 트랜잭션 및 데이터 정합성 흐름 (Saga & Outbox 패턴)

MSA 전환 시 가장 큰 기술적 난제인 **Database per Service 환경에서의 분산 트랜잭션**은 2PC(Two-Phase Commit)의 성능 병목을 피하기 위해 **Choreography 기반 Saga 패턴**과 **Transactional Outbox 패턴**을 표준으로 채택함.

Transactional Outbox 및 Choreography Saga 흐름




다이어그램 2. 주문 로컬 트랜잭션과 Outbox 원자적 커밋 및 결제 실패 시 자동 보상 트랜잭션(Compensation) 처리 흐름

### 4.3 MSA 핵심 구성요소별 표준 기술 스택 및 검증 내역

- **API Gateway & Routing ( `Spring Cloud Gateway` 또는 `Kong` )**
  - **역할**: 단일 진입점(Single Entry Point), JWT 통합 인증/인가, 카나리 라우팅(가중치 기반 트래픽 분할), Rate Limiting(Redis Token Bucket) 수행함.
  - **검증 수치**: 비동기 Netty 기반으로 단일 노드(2 vCPU, 4GB RAM)당 **4,500 RPS 처리 시 게이트웨이 부가 지연 4ms 미만** 확인됨 `[벤치마크 기준]`.
- **Resilience & Fault Tolerance ( `Resilience4j` )**
  - **역할**: 하위 서비스 응답 지연 시 스레드 풀 고갈을 막기 위한 **Circuit Breaker(서킷 브레이커)**, **Bulkhead(격벽)**, **TimeLimiter**, **Retry(지수 백오프)** 적용함.
  - **설정 기준**: 최근 100건 호출 중 실패율 **50% 이상** 또는 느린 호출(2초 초과) **80% 이상** 시 서킷 OPEN 전환 후 Fallback 응답 즉시 반환함.
- **비동기 메시징 및 이벤트 드리븐 ( `Apache Kafka` )**
  - **역할**: 서비스 간 시간적 결합(Temporal Coupling) 제거, 이벤트 소싱/CQRS 읽기 모델 동기화, 트래픽 스파이크 버퍼링(Backpressure 완화) 담당함.
  - **검증 수치**: 3-Broker 구성 시 초당 **50,000건 이상 메시지 처리** 및 End-to-End 이벤트 전달 지연 **평균 15ms 이내** 보장됨.
- **기술적 제약사항 (Constraints)**
  - **네트워크 직렬화 오버헤드**: 외부/BFF 통신은 `REST(JSON)`를 유지하되, 내부 서비스 간 고빈도 동기 호출 구간은 필요시 `gRPC(Protobuf)`를 적용하여 페이로드 크기를 **60% 이상 축소**해야 함.
  - **컨슈머 멱등성(Idempotency) 필수 구현**: Kafka의 `At-least-once` 전달 특성상 중복 메시지 수신이 발생할 수 있으므로, 모든 이벤트 컨슈머는 `Event_ID` 기반 멱등성 테이블 또는 Redis 분산 락 검증을 필수로 구현해야 함.

## 5. 주요 리스크 및 대응 방안

**요약**: MSA 전환 과정에서 발생 가능한 5대 핵심 리스크(기술/데이터/운영/일정/비용)를 식별하고 사전 예방책 및 비상 우회 계획(Contingency Plan)을 수립함.

| 구분 | 리스크 요인 | 발생 가능성 | 영향도 | 사전 예방 및 대응 방안 (Mitigation & Contingency) |
|---|---|---|---|---|
| **데이터** | 서비스 분리 후 분산 트랜잭션 실패로 인한 **주문-결제 간 데이터 불일치** 발생 | 중 | **상** | - **Transactional Outbox 패턴** 및 컨슈머 멱등성 키(Idempotency Key) 적용함<br>- **[우회안]** Dead Letter Queue(DLQ) 적재 및 5분 주기 정합성 대사(Reconciliation) 배치로 자동 보정 및 운영자 슬랙 알림 발송함 |
| **기술** | 서비스 간 동기 호출 체인(Call Chain) 심화로 인한 **연쇄 장애(Cascading Failure) 및 지연 증폭** | 중 | **상** | - 동기 호출 깊이를 **최대 2-Hop 이내**로 아키텍처 룰 강제함 (3-Hop 이상은 Kafka 비동기 이벤트로 전환)<br>- **[우회안]** `Resilience4j` 서킷 브레이커 작동 시 캐시된 기본값 반환 또는 대기열(Queue) 접수로 우회 처리함 |
| **운영** | 다수 마이크로서비스 분산 배포로 인한 **장애 원인 추적(Debugging) 시간 증가** | 상 | 중 | - 모든 요청 헤더에 `W3C Trace Context(traceparent)`를 강제 주입하고 **OpenTelemetry + Grafana Tempo/Loki**를 본 개발 전에 선행 구축함<br>- 단일 TraceID로 전 구간 로그·쿼리·메시지를 10초 내 조회 가능하도록 표준화함 |
| **일정** | 레거시 모놀리스 DB 테이블 강결합(Foreign Key/Join)으로 인한 **스키마 분리 지연** | 중 | 중 | - 초기에는 뷰(View) 또는 **Debezium CDC**로 레거시 DB 변경분을 신규 서비스 DB로 실시간 복제하여 읽기 전용으로 먼저 분리함<br>- **[우회안]** API Gateway 단에서 Strangler Fig 라우팅 비율을 조절하여 문제 발생 시 1분 내 레거시 모놀리스로 즉시 롤백(Fallback)함 |
| **비용** | 클라우드 관리형 서비스(K8s, MSK, 다중 RDS) 도입에 따른 **인프라 기본 고정비 상승** | 하 | 중 | - 개발/스테이징 환경은 단일 K8s 클러스터 내 Namespace 논리 분리 및 Spot Instance(최대 60% 할인)를 적극 활용함<br>- 월 예산 임계치(80%, $1,760) 도달 시 CloudWatch/Billing 알람 즉시 발송 설정함 |

---

## 6. 최종 권고안 및 향후 추진 일정

**요약**: 총괄 PL 승인 시 총 **8주(50 M/D)** 일정으로 4단계 점진적 전환을 추진하며, Phase 1(2주차) PoC 검증 완료 후 본 개발 전환 여부를 게이트 리뷰로 최종 확정함.

### 6.1 단계별 추진 로드맵

| 단계 | 주요 과업 (Action Items) | 소요 기간 (일정) | 핵심 산출물 및 완료 기준 (Exit Criteria) |
|---|---|---|---|
| **Phase 1: 아키텍처 설계 및 PoC 검증** | - 도메인 경계(Bounded Context) 도출 및 API 명세 확정<br>- API Gateway + 주문 서비스 프로토타입 + Kafka Outbox 연동 PoC<br>- K8s 개발 클러스터 및 OpenTelemetry 추적 파이프라인 구축 | **1주차 ~ 2주차 (2주 / 12 M/D)** | - **PoC 검증 보고서** (목표: 3,000 TPS, P95 100ms 이하 달성 확인)<br>- MSA 표준 아키텍처 및 이벤트 스키마 설계서 |
| **Phase 2: 핵심 도메인 분리 및 본 개발** | - 주문(Order) / 결제(Payment) / 알림(Notification) 서비스 분리 구현<br>- Saga 보상 트랜잭션, 멱등성 처리, Resilience4j 서킷 브레이커 적용<br>- 레거시 모놀리스 ACL(Anti-Corruption Layer) 및 데이터 마이그레이션 | **3주차 ~ 5주차 (3주 / 20 M/D)** | - 핵심 3개 마이크로서비스 운영 배포 가능 코드 (단위/통합 테스트 커버리지 80% 이상)<br>- 분산 트랜잭션 장애 주입(Chaos) 테스트 통과 |
| **Phase 3: 통합 관제 및 스테이징 부하 테스트** | - Grafana 대시보드(Golden Signals: 지연, 트래픽, 에러율, 포화도) 구축<br>- 피크 트래픽 1.5배(4,500 TPS) 부하 테스트 및 DB/K8s HPA 튜닝 | **6주차 ~ 7주차 (2주 / 10 M/D)** | - 부하 테스트 결과 보고서 및 튜닝 완료<br>- MSA 장애 대응 플레이북(Runbook) 및 운영 가이드 |
| **Phase 4: 카나리 배포 및 안정화** | - API Gateway 기반 트래픽 **10% → 50% → 100%** 단계적 전환 (Strangler Fig)<br>- 레거시 모놀리스 내 구(Old) 주문/결제 모듈 데드코드 정리 및 운영 이관 | **8주차 (1주 / 8 M/D)** | - **무중단 100% 전환 완료 보고서**<br>- 배포 리드타임 단축 확인 (4.5시간 → 15분 이내) |

### 6.2 참고 자료 (References)

- Microservices Pattern: Strangler Fig Application (Martin Fowler)
- Microservices.io: Pattern - Transactional Outbox & Saga (Chris Richardson)
- Spring Cloud Gateway & Resilience4j Official Reference Documentation
- OpenTelemetry Specification: W3C Trace Context for Distributed Tracing