전금법망 Sentry 플랫폼 구조 개선

외부 통신이 제한된 전금법망 환경에서, Sentry의 안정성·처리 성능·확장성을 개선하기 위한 구조 개선 프로젝트를 추진했습니다.

Kubernetes Helm Sentry Kafka PostgreSQL

프로젝트 배경

- 기존 Docker Compose 단일 노드 운영은 확장성·안정성이 부족해 트래픽 증가 시 서비스 중단과 로그 유실이 빈번했습니다.
- 또한 Kafka와 DB 구조의 한계로 처리 지연·부하 집중·장애 복구 지연 등 운영 효율 저하와 리스크가 지속되었습니다.

기존 환경 문제점

- 전금법망 제약: 외부 네트워크 차단으로 내부 모듈 에러가 반복되고, 에러 코드 확인·지표 수집이 제한
- 확장성 부족: 부하 증가 시 서비스가 자주 중단되고 리소스 한계에 빠르게 도달
- 로그 유실: 오류 로그 수집 실패가 반복되고, 트래픽 집중 시간대에는 로그가 처리되지 않음
- 연결 과부하: Kafka·PostgreSQL 간 연결 과부하로 처리 지연 및 장애 복구가 어려움
- 파티션 불균형: 특정 Kafka 파티션에 메시지가 집중되어 Consumer Lag이 급증, 로그 수집 성능 저하

설계 및 개선 과정

Kubernetes 기반 Helm 전환

- 사내 Kubernetes 인프라 자원과 HPA(수평 자동 확장)를 활용하여 안정적인 자원 운영 및 확장성 확보

Kafka 파티션 및 로그 수집 구조 최적화

- 특정 파티션으로 몰리던 메시지를 내부 코드 수정을 통해 균등 분산
- Sentry Worker 구조 개선으로 Consumer Lag 감소 및 로그 수집 성능 향상

전금법망 특성에 따른 모니터링 코드 개선

- 외부 네트워크 차단으로 인한 에러 코드 확인·지표 수집 제약을 해소
- 내부 환경에 독립적인 로그 수집 및 에러 코드 매핑 모듈 자체 구현

DB 성능 및 고가용성(HA) 확보

- PostgreSQL Pgpool 도입으로 Write/Read 트래픽 분산 처리
- pg_auto_failover 기반 HA 구성으로 장애 시 자동 승격 및 무중단 서비스 유지
- 백업 프로세스 구현

모니터링 체계 고도화

- Kafka 상태를 실시간으로 관리하기 위해 CMAK(Kafka Manager), Grafana 기반 모니터링 체계 구축

프로젝트 성과 및 기여

프로젝트 성과

- 이슈 생성 시간 단축 (평균 1시간 → 0초~1분 이내)
- 초당 처리량 향상 (3,000건 제한 → 12,000건 이상 처리)
- Kafka Lag 대폭 감소 (최대 2,800만 건 → 162만 건(94%↓))
- 부하에 따라 자동 확장되는 안정적·확장성 높은 운영 구조로 고도화

프로젝트 기여도

- Kubernetes(Helm) 전환: 온프레미스 Sentry를 사내 Kubernetes 기반 구조로 이전
- PostgreSQL 개선: 부하 분산 및 고가용성(HA) 구조 구현, 백업·복구 프로세스 설계 및 구축
- Kafka 최적화: 메시지 분배 로직 및 파티셔닝 개선, Consumer Offset 이슈 자동화
- Sentry 업그레이드 및 유지보수: 주요 버전 업그레이드, 이슈 및 문의 대응
- 문서화: 설계·운영·복구 가이드 등 체계적 운영 문서 작성 및 관리