실시간 금융 거래 탐지 시스템

RealFDS Architecture

프로젝트명	RealFDS (Real-time Financial Detection System)
버전	1.0
작성일	2025년 11월 06일
아키텍처	이벤트 기반 마이크로서비스 (5개 서비스)
목적	실시간 금융 거래 모니터링 및 사기 패턴 탐지

1. 시스템 개요

RealFDS는 Apache Kafka를 메시지 브로커로, Apache Flink를 스트림 처리 엔진으로 사용하여 금융 거래를 실시간으로 모니터링하고 의심스러운 패턴을 즉시 탐지하는 마이크로서비스 아키텍처 기반 시스템입니다. 거래 발생부터 알림 표시까지 5초 이내의 종단 간 지연 시간을 목표로 하며, Docker Compose를 사용하여 단일 명령어로 전체 시스템을 실행할 수 있습니다.

1.1 주요 기능

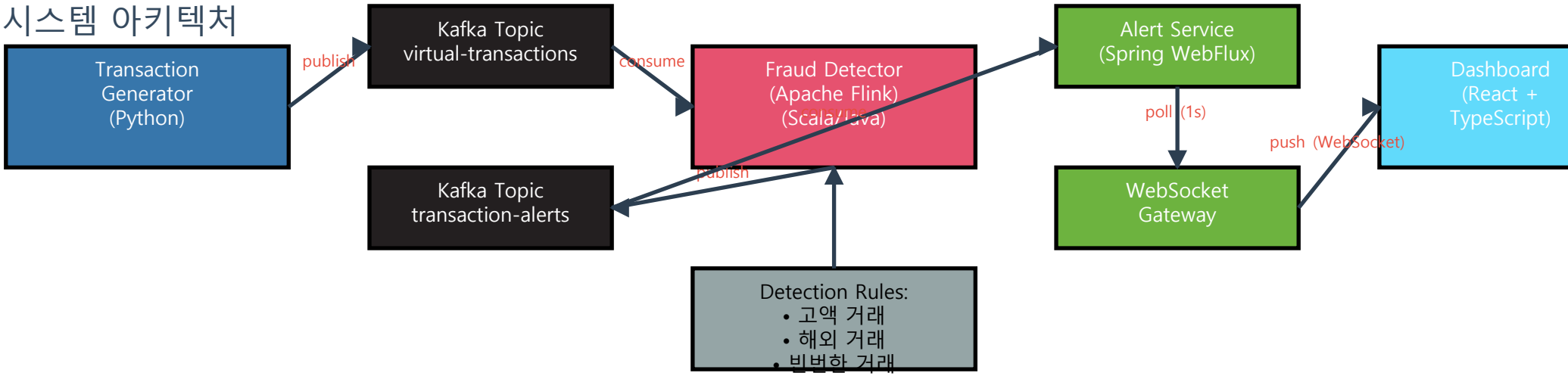
- 고액 거래 탐지: 100만원 초과 거래 자동 탐지
- 해외 거래 탐지: 한국 외 지역 거래 실시간 모니터링
- 빈번한 거래 탐지: 1분 내 5회 초과 거래 패턴 분석
- 실시간 알림: WebSocket을 통한 브라우저 실시간 푸시
- 간편한 실행: docker-compose up 단일 명령어로 전체 시스템 구동

1.2 기술 스택

구분	기술	용도
메시지 브로커	Apache Kafka 3.6+	서비스 간 이벤트 스트리밍
스트림 처리	Apache Flink 1.18+ (Scala)	실시간 거래 탐지 엔진
백엔드	Spring Boot 3.2+ WebFlux (Java 17)	알림 서비스 및 WebSocket 게이트웨이
프론트엔드	React 18 + TypeScript 5 + Vite	실시간 대시보드
데이터 생성	Python 3.11+	가상 거래 데이터 생성기
컨테이너	Docker + Docker Compose	전체 시스템 오케스트레이션

상태 관리	Flink RocksDB State Backend	빈번한 거래 탐지용 상태 저장

2. 시스템 아키텍처



범례:

Python | Flink | Spring | React | Kafka

성능 목표

- 종단 간 지연 시간: 평균 3초, 최대 5초
- 처리량: 초당 10개 거래 (목표), 초당 100개 (최대)
- 시스템 시작 시간: 5분 이내 | 가용성: 99%+

3. 컴포넌트 상세 설명

3.1 Transaction Generator (거래 생성기)

기술: Python 3.11+

테스트 및 데모를 위한 가상 거래 데이터를 주기적으로 생성합니다. 10명의 가상 사용자 풀에서 무작위로 선택하여 1,000원~1,500,000원 사이의 거래를 생성하고, Kafka virtual-transactions 토픽으로 발행합니다.

- 거래 금액, 국가 코드, 사용자 ID 무작위 생성
- 데모 효과를 위해 고액/해외 거래 포함
- Kafka Producer로 실시간 스트리밍

3.2 Fraud Detector (탐지 엔진)

기술: Apache Flink 1.18+ (Scala/Java)

Kafka에서 거래 스트림을 소비하고 3가지 탐지 규칙을 적용하여 의심스러운 패턴을 실시간으로 탐지합니다. 상태 기반(stateful) 처리를 통해 빈번한 거래 패턴을 추적합니다.

- HIGH_VALUE: 100만원 초과 거래 탐지
- FOREIGN_COUNTRY: KR 외 국가 거래 탐지
- HIGH_FREQUENCY: 1분 내 5회 초과 거래 탐지 (상태 관리)
- RocksDB State Backend로 사용자별 상태 유지
- Event-time 처리 및 Watermark 사용

3.3 Alert Service (알림 서비스)

기술: Spring Boot 3.2 WebFlux (Java 17)

Kafka transaction-alerts 토픽에서 알림을 소비하여 인메모리 저장소에 보관합니다. 최근 100개의 알림만 유지하며, REST API를 통해 조회를 제공합니다.

- Reactive Kafka Consumer (Spring Kafka)
- ConcurrentDeque를 사용한 인메모리 저장소
- REST API (/api/alerts) 제공
- Health check 엔드포인트

3.4 WebSocket Gateway (웹소켓 게이트웨이)

기술: Spring Boot 3.2 WebFlux (Java 17)

Alert Service를 1초마다 폴링하여 새로운 알림을 감지하고, 연결된 모든 WebSocket 클라이언트에게 실시간으로 브로드캐스트합니다.

- WebSocket 연결 관리 (Spring WebFlux WebSocket)
- Alert Service REST API 폴링 (1초 주기)
- 다중 클라이언트 브로드캐스트
- 자동 재연결 지원

3.5 Frontend Dashboard (프론트엔드 대시보드)

기술: React 18 + TypeScript 5 + Vite

보안 담당자가 웹 브라우저를 통해 실시간 알림을 모니터링할 수 있는 대시보드입니다. WebSocket을 통해 알림을 실시간으로 수신하고 화면에 표시합니다.

- WebSocket 클라이언트 (useWebSocket hook)
- 실시간 알림 목록 표시 (최근 100개)
- 연결 상태 표시 (connected/disconnected)
- 자동 재연결 (5초 간격)
- 반응형 UI

4. 데이터 흐름

- Transaction 생성 → virtual-transactions 토픽
- Flink 3가지 규칙 적용 (고액/해외/빈번)
- Alert 생성 → transaction-alerts 토픽
- Alert Service 저장 (최근 100개)
- WebSocket으로 브라우저 푸시 (1초 이내)

전체 흐름 시간: 거래 발생 → 화면 표시까지 평균 3초, 최대 5초

4.1 정상 흐름 시나리오

Step 1: Transaction Generator가 가상 거래 생성 (거래 ID, 사용자 ID, 금액, 국가 등)

Step 2: Kafka Producer가 virtual-transactions 토픽에 발행 (userId를 키로 파티셔닝)

Step 3: Fraud Detector (Flink)가 거래 소비 및 3가지 규칙 적용

Step 4: 조건 충족 시 Alert 생성 → transaction-alerts 토픽에 발행

Step 5: Alert Service가 알림 소비 및 인메모리 저장소에 추가 (최근 100개)

Step 6: WebSocket Gateway가 1초마다 Alert Service 폴링

Step 7: 새 알림 감지 시 연결된 모든 WebSocket 클라이언트에 브로드캐스트

Step 8: Frontend Dashboard가 알림 수신 및 React State 업데이트 → UI 렌더링

4.2 Kafka 토픽 구성

토픽명	용도	파티션	보관 기간	키
virtual-transactions	거래 스트림	3개	1시간	userId
transaction-alerts	알림 스트림	3개	24시간	userId

5. 탐지 규칙 상세

규칙명	유형	조건	심각도	사유 예시
HIGH_VALUE (고액 거래)	SIMPLE_RULE	amount > 1,000,000	HIGH	고액 거래 (100만원 초과): 1,250,000원
FOREIGN_COUNTRY (해외 거래)	SIMPLE_RULE	countryCode != "KR"	MEDIUM	해외 거래 탐지 (국가: US)
HIGH_FREQUENCY (빈번한 거래)	STATEFUL_RULE	1분 내 5회 초과	HIGH	빈번한 거래 (1분 내 5회 초과): user-3, 6회

참고: STATEFUL_RULE인 HIGH_FREQUENCY는 Flink의 RocksDB State Backend를 사용하여 사용자별 거래 이력을 1분 윈도우 내에서 추적합니다. Event-time 처리와 Watermark를 사용하여 정확한 시간 기반 집계를 수행합니다.

6. 성능 목표 및 제약사항

항목	목표	허용 최대	측정 방법
종단 간 지연 시간	평균 3초	p95 5초, 최대 8초	타임스탬프 차이 측정
처리량	초당 10개 거래	초당 100개	Kafka 메트릭
알림 표시 지연	1초 이내	2초	WebSocket 수신 → UI 렌더링
시스템 시작 시간	5분 이내	-	docker-compose up 완료
총 메모리 사용량	<4GB	-	모든 컨테이너 합계

시스템 가용성	99%	30분 무중단	안정성 테스트

6.1 제약사항

- 로컬 실행 전용: 클라우드 서비스 미사용, Docker Compose로만 실행
- 인메모리 저장소: 시스템 재시작 시 알림 이력 초기화
- 단일 브로커: Kafka 복제 계수 1 (고가용성 미지원)
- 최소 사양: 8GB RAM, 4 core CPU 권장
- 가상 데이터: 실제 금융 시스템 연동 없음 (데모/학습 목적)

7. 시스템 실행 방법

7.1 사전 요구사항

- Docker 20.10+ 설치
- Docker Compose 2.0+ 설치
- 최소 8GB RAM, 4 core CPU
- 포트 가용성: 9092 (Kafka), 8081 (Alert Service), 8082 (WebSocket), 8083 (Frontend)

7.2 실행 명령어

```
# 1. ■■■■ ■■■■■ ■■
cd RealFDS

# 2. ■■ ■■■ ■■
docker-compose up

# 3. ■ ■■■■■■ ■■■■ ■■
http://localhost:8083

# 4. ■■■ ■■
docker-compose down
```

참고: 시스템 시작 후 5분 이내에 모든 서비스가 준비되며, 대시보드에 접속하면 자동으로 실시간 알림 스트리밍이 시작됩니다.

8. 결론 및 향후 계획

RealFDS는 이벤트 기반 마이크로서비스 아키텍처의 핵심 개념을 실습할 수 있는 학습용 프로젝트입니다. Apache Kafka와 Apache Flink를 활용한 실시간 스트림 처리, WebSocket을 통한 양방향 통신, Docker를 사용한 컨테이너 오케스트레이션 등 현대적인 분산 시스템 기술을 경험할 수 있습니다.

8.1 향후 개선 사항

- 영속성 추가: PostgreSQL 또는 MongoDB를 사용한 알림 이력 저장
- 머신러닝 탐지: 과거 데이터 학습을 통한 이상 패턴 자동 탐지
- 동적 규칙 관리: 시스템 재시작 없이 웹 UI에서 새로운 탐지 규칙 추가
- 사용자 인증: 여러 보안 담당자의 개별 계정 및 역할 기반 접근 제어
- 알림 통계: 시간대별, 규칙별 알림 발생 추이 시각화
- 클라우드 배포: Kubernetes를 사용한 운영 환경 배포
- 모바일 지원: 반응형 UI 또는 모바일 앱 개발
