출입 통제 시스템: 3-Version 제품 라인업 전략

본 문서는 기존의 복잡한 5노드 HA(고가용성) 아키텍처를 기반으로, 고객의 다양한 요구사항(예산, 신뢰도, 기존 인프라)에 대응하기 위한 세 가지 제품 라인업 전략을 정의합니다.

각 버전은 판정 주체, 인프라 구성, 핵심 컴포넌트(Redis, Kafka) 채택 여부가 달라집니다.

1. 버전 1: ACU 사용 버전 (안정성/레거시 모델)

- **핵심 개념:** "Smart Controller" (Push 방식)
  - 기존 현장에 고가의 `ACU (Access Control Unit)` 하드웨어가 이미 설치되어 있거나, 오프라인 안정성이 최우선인 고객을 위한 버전입니다.
  - 중앙 서버는 정책(누가, 언제, 어디를)을 ACU에 미리 **Push**합니다.
  - 모든 실시간 판정은 현장의 **ACU가 로컬에서 직접 수행**합니다.
- **판정 흐름:** 리더기 → ACU (로컬 판정) → 문 열림 → (사후) 중앙 서버로 로그 전송
- **인프라 구성 (예시: 1-Node)**
  - **서버:** 1 노드 (관리, 정책 Push, 로그 수집용)
  - **Database: PostgreSQL** (필수)
  - **Cache (Redis): 불필요.** (중앙 서버가 실시간 Pull 판정을 하지 않음)
  - **Messaging (Kafka): 선택 (권장).**
- **주요 고려 사항:**
  - **Redis가 필요 없는 이유:** 실시간 판정 주체가 ACU이므로, 중앙 서버의 0.1초 응답을 위한 Redis 캐시가 필요 없습니다.
  - **Kafka를 권장하는 이유:** 수백 대의 ACU가 동시에 로그를 중앙 서버로 전송(Push)할 경우, 1노드 서버의 PostgreSQL이 쓰기(Write) 부하를 감당하지 못할 수 있습니다. Kafka를 로그 버퍼로 사용하면 서버의 안정성이 크게 향상됩니다.

2. 버전 2: SDAC 버전 (고가용성/프리미엄 모델)

- **핵심 개념:** "Dumb Terminal" (Pull 방식)

    - 저렴한 IP 리더기만 사용하며, 모든 판정 로직과 고가용성을 **중앙 서버**에 집중하는 풀 스펙(Full-Spec) 버전입니다.
    - 모든 리더기는 출입 시도 시 실시간으로 중앙 서버(Access Control Service)에 **Pull**하여 판정을 요청합니다.
- **판정 흐름:** 리더기 → API Gateway → Access Control Service (Redis 조회 판정) → 리더기로 응답 → 문 열림
- **인프라 구성 (예시: 3-Node 이상 HA)**
    - **서버:** 3 노드 HA (Active-Active)
    - **Database: PostgreSQL** (필수)
    - **Cache (Redis): 필수.** (0.1초 실시간 판정을 위한 핵심 캐시)
    - **Messaging (Kafka): 필수.** (판정 속도에 영향을 주지 않고 로그를 비동기 처리하기 위한 버퍼)
- **주요 고려 사항:**
    - 기존 10개 설계 문서의 최종 목표 모델입니다.
    - 서버 장애 시에도 서비스 중단이 없어야 하는 대규모 사업장, 데이터 센터 등에 적합합니다.
    - 서버 인프라 비용(고성능 서버, NVMe, RAM)이 높게 발생합니다.

3. 버전 3: 지능형 단말기 버전 (유연성/엣지 모델)

- **핵심 개념:** "Edge Controller" (Push 방식)
    - ACU라는 값비싼 하드웨어 대신, 정책 동기화(Push)가 가능한 **"지능형 IP 리더기"**를 사용하는 현대적인 버전입니다. (1번 버전의 변형)
    - 중앙 서버가 정책을 지능형 단말기에 **Push**합니다.
    - 실시간 판정은 **단말기(엣지)가 로컬에서 직접 수행**합니다.
- **판정 흐름:** 리더기 (로컬 판정) → 문 열림 → (사후) 중앙 서버로 로그 전송
- **인프라 구성 (예시: 1-Node 또는 HA)**
    - **서버:** 1 노드 (또는 고객 요구 시 HA 구성)
    - **Database: PostgreSQL** (필수)
    - **Cache (Redis): 불필요.** (중앙 서버가 실시간 Pull 판정을 하지 않음)

    - **Messaging (Kafka): 필수.** (수천 개의 엣지 단말기가 전송하는 로그/이벤트를 안정적으로 수집하기 위한 버퍼)
- **주요 고려 사항:**
    - ACU의 '안정성'과 SDAC의 '저렴한 하드웨어'의 장점을 결합한 모델입니다.
    - 단말기 펌웨어(S/W)가 정책 동기화, 로컬 판정, 오프라인 로깅 등 복잡한 기능을 지원해야 하는 기술적 과제가 있습니다.

요약: 개발 전략

이 3-Version 전략은 **`Access Control Service`**에 두 가지 핵심 로직을 모두 구현해야 함을 의미합니다.

1. **Pull Mode (버전 2용):** Redis 캐시를 조회하여 실시간 판정 응답을 반환하는 로직.
2. **Push Mode (버전 1, 3용):** PostgreSQL의 정책을 ACU 또는 지능형 단말기로 동기화(Push)하는 로직.

다른 서비스(`User`, `Policy`, `Log` 등)는 3가지 버전에서 대부분 공통으로 재사용될 수 있습니다.