출입 통제 시스템: 서비스 아키텍처 정의서

본 문서는 출입 통제 시스템을 구성하는 마이크로서비스(MSA) 및 백엔드 인프라의 역할과 책임을 상세하게 정의합니다.

1. 애플리케이션 계층 (Spring Cloud Services)

시스템의 핵심 비즈니스 로직을 담당하는 Spring Cloud 기반의 마이크로서비스입니다.

A. API Gateway (API 게이트웨이)

- **역할:** 시스템의 유일한 관문(Entry Point) 및 보안 초소.
- **핵심 기능:**
  - **인증/인가:** 모든 수신 요청을 검사합니다.
    - 관리자 요청: `JWT Access Token`을 검증하고 토큰 내의 권한(Role)을 확인합니다.
    - 단말기(SDAC) 요청: `Device Token`을 검증하여 인가된 하드웨어인지 확인합니다.
  - **라우팅:** 요청 경로(예: `/api/admin/users/**`)를 분석하여 적절한 내부 서비스(예: `User Service`)로 중계합니다.
  - **장애 전파 차단:** `User Service` 등 내부 서비스가 응답하지 않을 경우, Circuit Breaker를 작동시켜 시스템 전체의 마비를 방지합니다.

B. Access Control Service (출입 판정 서비스)

- **역할:** 시스템의 "두뇌". 모든 실시간 출입 판정 및 비상 상황을 제어합니다.
- **핵심 기능:**
  - **실시간 판정 (SDAC):** 0.1초 내 판정을 위해 `Redis` 캐시를 조회하여 `GRANT` / `DENY`를 결정합니다.
  - **고급 보안 로직:** '안티-패스백', '문 열림 시간 초과', '강제 개방' 등 보안 규칙을 처리합니다.
  - **비상 제어:** '화재 연동', '전체 봉쇄(Lockdown)' 명령을 최우선으로 처리합니다.

- **정책 동기화 (ACU):** 관리자가 정책 변경 시, `ACU` 하드웨어로 접속하여 최신 정책을 `Push` 합니다.
- **비동기 로깅:** 판정 완료 즉시, 로그 데이터를 `Kafka`로 발행(Produce)합니다. (로그 저장이 판정 속도에 영향을 주지 않도록 분리)

C. User Service (출입자 및 권한 관리)

- **역할:** 시스템의 모든 '개인(Identity)' 정보 및 '인가(Authorization)' 정책을 관리합니다.
- **핵심 기능:**
  - **출입자 관리:** 출입자(직원/방문객) 정보, 인증 매체(카드/지문/QR)를 생성(Create), 조회(Read), 수정(Update), 삭제(Delete)합니다.
  - **관리자 계정 관리:** Admin UI에 로그인하는 관리자 계정을 관리합니다.
  - **RBAC (권한 관리):** '역할(Role)'과 '권한(Permission)'을 정의하고 관리자에게 매핑합니다.
  - **캐시 동기화:** 출입자 정보나 상태(예: '분실')가 변경되면, `PostgreSQL` 저장과 동시에 `Redis` 캐시를 갱신하여 `Access Control Service`가 항상 최신 정보를 보게 합니다.

D. Policy Service (정책 및 장치 관리)

- **역할:** 시스템의 모든 '물리적 환경'과 '접근 규칙'을 관리합니다.
- **핵심 기능:**
  - **환경 정의:** '서버실' 같은 `구역(Zone)`, '서버실 정문' 같은 `출입문(Door)`을 정의합니다.
  - **장치 관리:** `IP 리더기`, `ACU` 등 하드웨어를 등록하고 네트워크 정보를 관리합니다.
  - **규칙 관리:** '주간 근무 시간' 같은 `시간표(Schedule)`를 정의하고, "[A그룹]은 [B구역]을 [C시간표]에" 통과할 수 있다는 `접근 규칙(Access Rule)`을 매핑합니다.
  - **캐시 동기화:** 접근 규칙이 변경되면 `Redis` 캐시를 갱신하여 실시간 판정에 반영합니다.

E. Auth Service (인증 서비스)

- **역할:** 관리자(Admin UI)의 '인증(Authentication)'을 전담합니다.
- **핵심 기능:**
  - **토큰 발급:** 관리자 로그인 성공 시, API 호출용 `Access Token`과 갱신용 `Refresh Token`을 발급합니다.

- **토큰 갱신:** `Access Token` 만료 시, `Refresh Token`을 받아 새 `Access Token`을 재발급합니다.
- **로그아웃:** 토큰을 무효화(Revoke)하여 세션을 종료시킵니다.

F. Log Service (로그 및 통계 서비스)

- **역할:** 시스템의 모든 감사 로그를 수집, 저장하고 통계 데이터를 제공합니다.
- **핵심 기능:**
  - **로그 수신 (Kafka Consumer):** `Kafka`의 출입 로그 및 관리자 활동 로그 토픽을 구독(Consume)합니다.
  - **영구 저장:** 수신한 로그를 `PostgreSQL`의 로그 테이블에 저장합니다.
  - **이중 적재 (검색 성능):** (개선안) 수신한 로그를 `PostgreSQL`(영구 보관용)과 `Elasticsearch`(빠른 검색 및 통계용)에 이중으로 적재합니다.
  - **API 제공:** Admin UI의 '감사 로그 검색' 및 '통계' 요청에 따라 `PostgreSQL`에서 데이터를 조회하여 반환합니다.
  - **API 제공:** Admin UI의 '감사 로그 검색' 및 '통계' 요청 시, `Elasticsearch`를 우선 조회하여 DB 부하 없이 빠른 응답을 제공합니다. (데이터는 `PostgreSQL`에서 조회할 수도 있음)

2. 데이터, 관리 및 인프라 계층 (Backing & Management Services)

애플리케이션 계층이 원활하게 동작하도록 지원하는 필수 기반 서비스입니다.

A. PostgreSQL Cluster (with Patroni)

- **역할:** 모든 설정(사용자, 정책) 및 로그 데이터의 **영구 저장소(Database)**입니다.
- **HA:** Patroni가 데이터베이스 서버 장애 시 자동으로 Standby 서버를 Primary로 승격시켜 중단 없는 서비스를 보장합니다.

B. Redis Cluster

- **역할:** `Access Control Service`의 0.1초 미만 판정을 위한 **초고속 인메모리 캐시(Cache)**입니다.
- **핵심 기능:** `User Service`와 `Policy Service`가 동기화한 데이터를 메모리에 저장하여 DB 접근 없이 즉각적인 조회를 지원합니다.

C. Kafka Cluster

- **역할:** 판정 속도와 로그 저장 속도를 분리(Decoupling)하는 **비동기 메시지 버퍼(Buffer)**입니다.
- **핵심 기능:** `Access Control Service`는 로그를 Kafka에 '발행'만 하고 즉시 다음 판정을 준비하며, `Log Service`는 Kafka에서 로그를 '구독'하여 천천히 DB에 저장합니다.

D. Docker Swarm

- **역할:** 모든 마이크로서비스 컨테이너의 **실행 플랫폼(Orchestrator)**입니다.
- **HA:** 3-노드 클러스터로 구성되어, 특정 서버(노드) 장애 시 해당 서버의 컨테이너를 다른 정상 서버에 자동으로 재시작시켜 서비스 연속성을 보장합니다.

E. Spring Cloud Config Server

- **역할:** 중앙 집중식 설정 관리.
- **핵심 기능:** 모든 마이크로서비스(Gateway, User, Policy 등)의 `application.yml` 설정을 중앙에서 관리합니다. DB 접속 정보, JWT 비밀 키 등을 Git 등과 연동하여 보관하며, 서비스 재배포 없이 동적 갱신을 지원합니다.

F. Internal HA DNS

- **역할:** 클러스터 외부의 고가용성(HA) 접속 주소 제공.
- **핵심 기능:** `api.sdac.local` 같은 단일 도메인 주소를 3대의 App 노드(Node 1, 2, 3) IP로 매핑합니다. Admin UI, IP 리더기 등 외부 클라이언트가 노드 1대 장애와 상관없이 항상 서비스에 접속할 수 있도록 보장합니다.

G. Admin UI Web Server (Nginx)

- **역할:** 관리자 웹 애플리케이션(UI) 제공.
- **핵심 기능:** React, Vue 등으로 빌드된 정적 파일(HTML, JS, CSS)을 관리자의 웹 브라우저로 전송(serve)하는 경량 웹 서버입니다.

H. WebSocket Service (실시간 이벤트 브릿지)

- **역할:** Kafka의 이벤트를 Admin UI로 실시간 중계합니다.
- **핵심 기능:** `Log Service`와 별개로, `Kafka`의 실시간 출입/알람 토픽을 구독(Consume)하여, 이를 즉시 Admin UI(웹 브라우저)로 `Socket.IO`를 통해 브로드캐스트합니다.

I. Monitoring Stack (Prometheus + Grafana)

- **역할:** 시스템 성능 및 상태 모니터링.
- **핵심 기능:** `Prometheus`가 모든 서비스와 노드의 CPU, RAM, 응답 속도, Circuit Breaker 상태 등 지표를 수집(Scrape)하고, `Grafana`가 이를 대시보드로 시각화하여 장애 예측 및 HA 예비 자원(Headroom)을 확인합니다.

J. Logging Stack (EFK / ELK)

- **역할:** 중앙 집중식 시스템/장애 로그 수집.
- **핵심 기능:** `Fluentd`(또는 Filebeat)가 모든 Docker 컨테이너의 표준 출력(stdout/stderr) 로그(예: Error Stack Trace)를 수집하여 `Elasticsearch`로 전송합니다. 개발자는 `Kibana`를 통해 장애 원인을 신속히 검색하고 분석합니다. (`Log Service`의 '비즈니스 로그'와 분리됨)

K. Cluster GUI (Portainer)

- **역할:** Docker Swarm 클러스터 시각적 관리.
- **핵심 기능:** GUI를 통해 Swarm 클러스터의 노드, 서비스, 컨테이너 상태를 모니터링하고 간단한 제어를 수행합니다.

L. MinIO (Object Storage)

- **역할:** 파일/객체 저장소.
- **핵심 기능:** `User Service`와 연동하여 출입자 및 관리자의 프로필 사진(photo_url)을 저장하고, Presigned URL을 통해 안전한 업로드를 지원합니다.