🚨 Infrastructure Specification: AI-Powered Accident Reporting Platform

เอกสารฉบับนี้สรุปโครงสร้าง Infrastructure สำหรับระบบแจ้งเหตุฉุกเฉินอัจฉริยะ (POC) ที่เน้นความปลอดภัยสูงสุดด้วยการเข้ารหัสข้อมูล และการทำงานแบบ Event-driven Microservices

1. High-Level Architecture Overview

ระบบใช้โครงสร้างแบบ **Microservices** เพื่อแยกภาระงาน (Workload) และความรับผิดชอบออกจากกัน โดยมี **Proxy Gateway** เป็นจุดศูนย์กลางในการจัดการความปลอดภัยก่อนกระจายงานผ่าน **Message Broker**

2. Infrastructure Components (Tech Stack)

A. Frontend Layer (Next.js)

- **Next.js (App Router):** ใช้เป็น Base Framework สำหรับทุกหน้าบ้าน
- **Subdomains:**
  - `app.yourdomain.com`: สำหรับ User แจ้งเหตุ (เน้น PWA/LIFF)
  - `rescue.yourdomain.com`: สำหรับอาสากู้ภัย (Map & Real-time alerts)
  - `admin.yourdomain.com`: สำหรับ Admin (Management & Logs)
- **Styling:** Tailwind CSS + ShadcnUI
- **Validation:** Zod (Shared Schema)

B. Gateway & Authentication (ElysiaJS)

- **Proxy Server:** ทำหน้าที่เป็น Reverse Proxy และ Authentication Guard
- **Auth Mechanism:**
  - **Encryption:** AES-256-GCM (เพื่อปกปิดข้อมูลใน Payload)
  - **Signature:** Ed25519 (Asymmetric Key เพื่อยืนยันแหล่งที่มา)
- **Rate Limiting:** ป้องกันการสแปมแจ้งเหตุปลอม

C. Core Microservices (Internal)

1. **Reporting Service:** จัดการการรับข้อมูลแจ้งเหตุ พิกัด และ Metadata
2. **AI Analysis Worker:** * Consumer ดึงงานจาก RabbitMQ
   - เชื่อมต่อกับ **OpenRouter** (วิเคราะห์รูปภาพผ่าน GPT-4o/Claude 3.5)
3. **Socket Service:** จัดการ WebSockets แบบรวมศูนย์ เพื่อ Push ข้อมูล Real-time
4. **Storage Service:** จัดการการ Upload/Download กับ **MinIO (Self-hosted S3)**

D. Data & Middleware Layer

- **Database:** MongoDB (เก็บข้อมูล Report และ AI Analytics)

- **Message Broker:** RabbitMQ (ตัวกลางรับส่งงานระหว่าง Service)
- **Cache:** Dragonfly (จัดการ Session และ WebSocket state)
- **Logging:** Axiom (Centralized Log Management)

3. Data Flow & AI Processing

ขั้นตอนการทำงาน (Lifecycle of a Report)

1. **User Submission:** User ส่งพิกัด + รูปภาพ -> ผ่านการตรวจสอบ Auth ที่ Proxy -> เข้าสู่ Reporting Service
2. **Image Upload:** รูปภาพถูกเก็บไว้ที่ **MinIO** และสร้าง Metadata ใน MongoDB
3. **Queue Injection:** Reporting Service ส่ง Message เข้าไปใน **RabbitMQ** (Queue: `incident_analyze`)
4. **AI Analysis:** AI Worker ดึงงานไปประมวลผลผ่าน OpenRouter โดยส่งรูปภาพไปวิเคราะห์ความรุนแรง (Severity) และประเภทอุบัติเหตุ
5. **User Confirmation:** ผลลัพธ์จาก AI ถูกส่งกลับหา User ผ่าน **WebSocket** เพื่อให้กดยืนยัน (Confirm) ข้อมูล
6. **Emergency Dispatch:** เมื่อยืนยัน ระบบจะแจ้งเตือนไปยังหน่วยกู้ภัยที่อยู่ใกล้เคียงทันที

4. Security & Encryption Strategy

ส่วนสำคัญ	วิธีการปกป้องข้อมูล
Authentication Token	เข้ารหัส AES-256-GCM และเซ็นชื่อด้วย EdDSA (Private Key)
Sensitive Data (DB)	ข้อมูล PII ใน MongoDB เช่น เบอร์โทร จะถูกเข้ารหัสแบบ AES (Field-level)
Object Storage	รูปภาพบน MinIO เป็นแบบ Private ต้องเข้าถึงผ่าน Pre-signed URL เท่านั้น
Internal Traffic	การสื่อสารระหว่าง Microservices ทำในวง Network ปิด (Docker Network)
Logging	Axiom จะเก็บ Correlation-ID เพื่อให้สามารถ Trace ข้อมูลได้แต่จะ Mask ข้อมูลสำคัญไว้

5. Deployment Recommendation (POC)

สำหรับการทดสอบช่วงแรก (POC) แนะนำให้ใช้ **Docker Compose** บน **Dedicated Server (Ubuntu)**:

- แยก Container สำหรับแต่ละ Microservice เพื่อความอิสระ
- ใช้ **Nginx หรือ Caddy** เป็นด่านหน้าจัดการ SSL Certificate
- กำหนด Resource Limit ให้กับ AI Worker เพื่อไม่ให้กิน CPU ของระบบหลักเมื่อมี Traffic เข้ามาจำนวนมาก

6. Monitoring (Axiom Integration)

ระบบจะส่ง Log แบบ Structured JSON ไปที่ Axiom ดังนี้:

- `level` : info, error, critical
- `service` : ชื่อ service ที่ส่ง log
- `correlation_id` : ID ประจำ Request เพื่อติดตามการทำงานข้าม Service
- `payload` : ข้อมูลที่เกี่ยวข้อง (โดยตัดข้อมูลลับออกแล้ว)