RCA Triage Bot (IATS) - One-Page App Summary

What it is

Incident Auto-Triage Service (IATS) is a prototype that ingests CloudWatch alarm events, collects supporting evidence, and generates a structured triage report. The stack runs locally with Docker Compose using a FastAPI backend, Celery worker, PostgreSQL, Redis, Ollama/OpenAI, and a React UI.

Who it's for

Primary user/persona: **Not found in repo.** (Workflow evidence suggests on-call SRE or platform engineers responding to CloudWatch alarms.)

What it does

- Accepts CloudWatch alarm events via `POST /v1/alerts/cloudwatch` and normalizes to a canonical `AlertEvent`.
- Computes a dedup key and upserts incidents so repeat alerts update an existing incident record.
- Queues async triage via Celery/Redis and transitions incident states (open -> triaging -> triaged/failed).
- Collects evidence from CloudWatch Logs Insights (fixture mode supported) and stores an evidence pack.
- Extracts local repo snippets by keyword to add code context into the evidence set.
- Generates strict JSON triage reports with either local Ollama or OpenAI, validates schema, and persists results.
- Exposes UI/API views for incident list, incident detail, evidence artifacts, and triage report output.

How it works (repo-evidenced architecture)

- **Ingress:** CloudWatch payload -> FastAPI route -> normalization + service registry lookup + dedup/upsert in PostgreSQL.
- **Async orchestration:** Ingestion enqueues Celery task in Redis; worker runs the triage pipeline.
- **Evidence layer:** Worker queries CloudWatch Logs adapter (or fixture), derives patterns, and adds local repo snippets.
- **Storage:** Alert events, incidents, evidence packs, and triage reports persisted via SQLAlchemy repositories.
- **Reasoning + output:** LLM adapter (local/OpenAI) returns schema-validated report; notifier emits console and optional Slack updates.
- **Presentation:** React UI reads backend endpoints (`/v1/incidents`, detail, evidence, report) to render triage results.

How to run (minimal getting started)

- Start stack: `docker compose -f infra/docker-compose.yml up --build`.
- Post sample alarm: `curl -sS -X POST http://localhost:8000/v1/alerts/cloudwatch -H "Content-Type: application/json" --data @fixtures/cloudwatch_alarm_event.json`.
- Open UI at `http://localhost:5173` (API health: `http://localhost:8000/health`).

Note: CloudWatch-only input source, explicit authentication/authorization model, and production deployment guidance are **Not found in repo**.