Distributed Weather App - System Report

Generated: 2025-11-27T22:41:31.265791Z

Overview

Edge nodes publish observations to MQTT. Aggregator ingests to Timescale, runs periodic aggregates, evaluates alert rules, and exposes API. Alerts are also published to MQTT. Frontend fetches API data and subscribes to MQTT alerts over WebSockets.

Infrastructure

docker-compose runs mosquitto (MQTT+WS 1883/9001), timescale/postgres (5432), and FastAPI aggregator (8080). init SQL defines hypertables observations/aggregates/alerts. mosquitto.conf enables external access + websockets.

Backend Services

FastAPI (app/main.py) endpoints: /health, /ready, /metrics, /cities/{id}/observations, /aggregates, /alerts, /forecast. mqtt_client subscribes to city/+/observations. ingest_worker validates and stores. aggregator_worker recomputes 15m/1h aggregates and runs alert rules. alerts.py stores + publishes alerts to alerts/{city}. forecast.py uses OpenWeather with caching (fallback placeholder). openweather_publisher.py polls OpenWeather current conditions and publishes to MQTT. config.py holds env settings (API_KEY, DB, MQTT, forecast, windows, CORS, OpenWeather poll/cities).

Data Contracts

Observation MQTT: {city_id, source, observed_at ISO, temp_c, humidity, wind_kph, pressure_hpa, rain_mm}. Alerts: {city_id, level, rule, message, triggered_at}. Aggregates: city_id, bucket_start, bucket_width, temp_avg/min/max, humidity_avg, wind_avg.

Alerting

Default rules: temp_max > 35C (1h), wind_avg > 40 kph (1h), humidity_avg > 0.85 (15m). Alerts stored in DB and published to MQTT alerts/{city}. Frontend subscribes live via mqtt.js and WS 9001.

Frontend

Static dashboard (index.html/style.css/script.js). Fetches API with X-API-Key. Connects to ws://localhost:9001 and subscribes to alerts/{city}. Shows observations, aggregates (15m), 3-day forecast, metrics, and live alerts.

Running the Stack

1) docker compose -f infra/docker-compose.yml up -d --build. 2) Serve frontend: cd frontend && python -m http.server 3000. 3) Open http://localhost:3000. 4) Publish test data via edge-sim/publisher.py or mosquitto_pub to city/{city}/observations. 5) Check API with curl -H 'X-API-Key: devkey' http://localhost:8080/health.

Configuration

Key env vars: API_KEY (X-API-Key), FORECAST_API_KEY/BASE_URL, MQTT_HOST/PORT/USERNAME/PASSWORD, AGGREGATE_INTERVAL_SECONDS, ALERT_COOLDOWN_MINUTES, OPENWEATHER_POLL_SECONDS, OPENWEATHER_CITIES (JSON),

ALLOWED_ORIGINS.

Files of Interest

infra/docker-compose.yml, infra/sql/init.sql, infra/mosquitto.conf, aggregator/app/*.py (main, mqtt_client, aggregator, alerts, forecast, openweather_publisher, schemas, config, db), frontend/index.html/style.css/script.js, edge-sim/publisher.py, start.py.

Notes

MQTT topics: city/{city}/observations, alerts/{city}. Timescale hypertables: observations(observed_at), aggregates(bucket_start). OpenWeather worker requires FORECAST_API_KEY set. Frontend uses default API key devkey (change for prod).