Candidate Exercise: Weather API with FastAPI

Task
Build a simple weather API service using FastAPI that fetches weather data from an external public API (e.g., OpenWeatherMap).
The service should use asynchronous programming and demonstrate cloud-style storage/logging, with local equivalents acceptable.

You should aim to spend 4–6 hours max.

Core Requirements

1. FastAPI Setup
- One endpoint: /weather
- Accepts a GET request with a city query parameter

2. Asynchronous Data Fetching
- Use Python’s asyncio/httpx to fetch weather data
- Handle errors (invalid city, API failure, etc.) gracefully

3. Storage (S3 or Local Equivalent)
- Save each weather response as a JSON file
- Filename: {city}_{timestamp}.json
- Local equivalent: write to a /data folder

4. Event Logging (DynamoDB or Local Equivalent)
- Log city, timestamp, and file path/URL
- Local equivalent: SQLite or JSON log file

5. Caching (5-Minute Expiry)
- Check for cached data before calling the API
- If available and <5 min old, return it

6. Deployment
- Provide Dockerfile and docker-compose.yml to run locally

Evaluation Criteria
- Correctness: Service works as specified
- Code Quality: Clean, modular, and readable code
- Async: Proper use of async features
- Storage & Logging: Simple abstractions (easy to swap to AWS later)
- Error Handling: Graceful failure modes
- Deployment: Easy to run locally with Docker

Bonus Challenges (Optional, for extra points)
- Unit/Integration Tests: Use pytest or equivalent
- CI/CD Setup: GitHub Actions or similar to run tests on push
- Rate Limiting: Prevent abuse by limiting requests per client
- Config Management: Externalize API keys and settings using .env or config files
- API Docs Polish: Ensure FastAPI’s autogenerated docs (/docs) are clean and descriptive

- Extend API: Add support for 3-day forecast or multiple cities in one request