Master 2 Data Science – Institut Polytechnique de Paris

Data Stream Processing

StormWatch AI

A Real-Time Weather Advisory System

Nardi Xhepi

nardi.xhepi@polytechnique.edu

Augustin Bresset

augustin.bresset@ip-paris.fr

December 2025

Abstract

Modern weather applications increasingly demand intelligent, conversational interfaces capable of answering complex natural language queries about atmospheric conditions. This report presents **StormWatch AI**, a real-time weather advisory system that integrates Apache Kafka for stream processing, Qdrant vector database for semantic storage, and a Retrieval-Augmented Generation (RAG) pipeline for intelligent query answering.

The system ingests weather data from the OpenWeatherMap API and French weather news from Le Monde RSS feeds, processes them through a distributed streaming architecture, generates semantic embeddings using sentence transformers, and provides natural language responses via Groq's LLM inference API.

The project can be found on GitHub at `https://github.com/nardi-xhepi/stormwatch-ai`.

Key Features

- Real-time weather data streaming with automatic API fallback
- Type-specific retrieval strategies for structured and unstructured data
- Natural language interface for weather queries and advisories

Contents

1 Introduction 4

2 External Services and Data Sources 5
2.1 Weather Data: OpenWeatherMap API 5
2.2 LLM Provider: Groq 5

3 Background: Core Technologies 6
3.1 Apache Kafka and Stream Processing 6
3.2 Vector Embeddings and Similarity Search 6
3.3 Retrieval-Augmented Generation (RAG) 6
3.4 Differentiated Retrieval Strategy 6

4 System Architecture 8
4.1 Data Ingestion Layer 8
4.2 Stream Processing Layer 8
4.3 News Integration 9

5 Application Interface 10
5.1 Interface Components 10

6 Implementation Details 11
6.1 Embedding Model Selection: Cross-Lingual News Retrieval 11
6.1.1 Benchmark Methodology 11
6.1.2 Cross-Lingual Retrieval Results 11
6.1.3 Trade-off Analysis 12
6.2 Type-Specific Retrieval Strategies 12
6.2.1 Deterministic Retrieval (Weather Data) 12
6.2.2 Semantic RAG with Temporal Decay (News Articles) 12
6.3 Location Filtering with Payload Indexes 13

7 Conclusion 14

1 Introduction

Weather information plays a critical role in daily decision-making, influencing activities ranging from transportation planning to agricultural operations. Traditional weather applications provide static displays of meteorological data (temperature, humidity, wind speed), but contemporary users increasingly expect more sophisticated interfaces. They want to ask natural language questions like *"Is it safe to drive today?"* or *"Should I bring an umbrella?"* and receive contextual, actionable insights rather than raw data.

The emergence of large language models (LLMs) and vector databases has created new opportunities for building conversational weather systems. However, integrating these technologies with real-time data streams presents significant architectural challenges. Weather data must flow continuously from external APIs, undergo processing and semantic encoding, and remain available for rapid retrieval during query processing. The system must handle the inherent tension between data freshness and query latency.

Our project addresses these challenges through **StormWatch AI**, a system that unifies three technological paradigms:

1. **Stream Processing** via Apache Kafka ensures continuous data freshness with fault-tolerant message delivery
2. **Vector Databases** enable semantic similarity search across weather observations
3. **Retrieval-Augmented Generation** provides intelligent responses grounded in current conditions

2 External Services and Data Sources

StormWatch AI integrates with two external cloud services: **OpenWeatherMap** for meteorological data and **Groq** for natural language generation.

2.1 Weather Data: OpenWeatherMap API

OpenWeatherMap [4] is a widely-used weather data provider offering global coverage through a RESTful API. The free tier provides access to several endpoints:

Table 1: OpenWeatherMap API Endpoints

Endpoint	**URL**	**Description**
One Call 3.0	`/data/3.0/onecall`	Primary: current, forecast, alerts (rate-limited)
Current Weather	`/data/2.5/weather`	Fallback: real-time observations
5-Day Forecast	`/data/2.5/forecast`	Fallback: 3-hour interval predictions
Air Quality	`/data/2.5/air_pollution`	Always used: PM2.5, PM10, AQI (separate quota)

The system primarily uses the One Call 3.0 API which bundles current weather, forecasts, and alerts in a single request. When rate limits are exceeded, the producer automatically falls back to the legacy 2.5 API endpoints for current conditions and forecasts. Air quality is always fetched separately via the 2.5 API, which has independent rate limits and continues working even when One Call is blocked.

The system monitors seven French metropolitan areas selected for geographic diversity: Lyon, Paris, Marseille, Toulouse, Nice, Rouen, and Bordeaux. Additional cities can be configured via the `MONITORED_CITIES` environment variable. The producer polls each city every 30 seconds (configurable), yielding approximately 2,400 observations per hour across all data types.

2.2 LLM Provider: Groq

For natural language generation, StormWatch AI utilizes **Groq** [5], distinguished by its custom **LPU** (Language Processing Unit) hardware achieving significantly higher throughput than GPU-based inference.

Table 2: Groq Model Configuration

Parameter	**Value**
Model	`openai/gpt-oss-120b` (120B parameters)
Temperature	0.3 (low creativity for factual responses)
Max Tokens	1024
Latency	1-3 seconds (50-100 tokens/second)

3 Background: Core Technologies

3.1 Apache Kafka and Stream Processing

Apache Kafka [1] is a distributed event streaming platform. Unlike traditional message queues that delete messages after consumption, Kafka persists messages with configurable retention periods, enabling multiple independent consumers to process the same stream.

Kafka organizes data into **topics**, each representing a logical category of messages. In our architecture, Kafka serves as the central nervous system connecting data ingestion to storage with four topics: `weather-data`, `weather-forecast`, `weather-alerts`, and `weather-news`.

Delivery Guarantees

StormWatch AI uses `acks=all` configuration, ensuring no weather observation is lost by waiting for all replicas to acknowledge before confirming delivery.

3.2 Vector Embeddings and Similarity Search

Traditional databases excel at exact-match queries but struggle with semantic similarity. Vector databases operate on high-dimensional embeddings that capture semantic content.

Given text t, an embedding model f_θ produces:

$$\mathbf{v} = f_\theta(t) \in \mathbb{R}^d$$

Cosine Similarity measures the angle between vectors:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} = \frac{\sum_{i=1}^{d} u_i v_i}{\sqrt{\sum_{i=1}^{d} u_i^2} \sqrt{\sum_{i=1}^{d} v_i^2}}$$

Vector databases like Qdrant [2] index embeddings using Approximate Nearest Neighbor (ANN) algorithms, achieving logarithmic search complexity rather than linear scanning.

3.3 Retrieval-Augmented Generation (RAG)

Large language models suffer from two limitations: **knowledge cutoffs** and **hallucination**. RAG [3] addresses these by grounding outputs in retrieved evidence through three phases:

1. **Retrieval**: Compute query embedding and search for similar documents
2. **Augmentation**: Format retrieved documents into context and construct prompt
3. **Generation**: Feed augmented prompt to LLM for response

3.4 Differentiated Retrieval Strategy

Rather than applying a single RAG approach to all data types, StormWatch AI implements type-specific retrieval strategies:

Table 3: Type-specific retrieval strategies

Data Type	**Method**	**Rationale**
Current Weather	Filter + Sort	Always need latest ground truth
Air Quality	Filter + Sort	Real-time data essential
Forecasts	Timestamp Sort	Return last 2-3 entries
Alerts	Validity Check	start $\leq$ now $\leq$ end
News Articles	Semantic RAG	Match on meaning, not keywords

4 System Architecture

StormWatch AI employs a layered architecture separating concerns of data ingestion, stream processing, semantic storage, and application logic. Figure 1 illustrates the design.

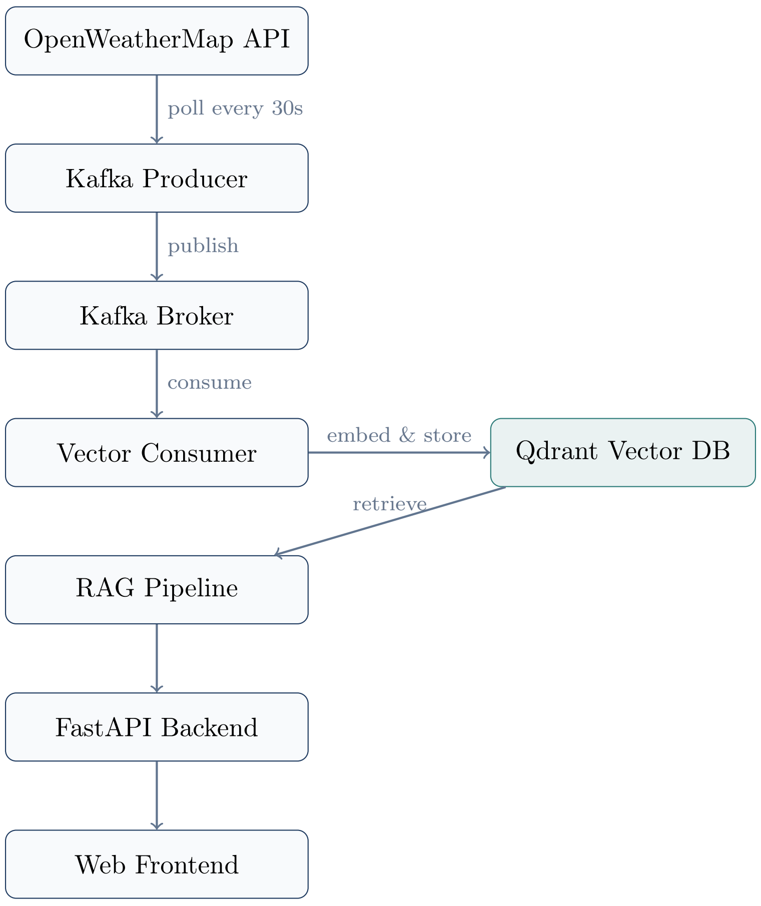


Figure 1: System architecture showing data flow from API ingestion through streaming, storage, and presentation layers.

4.1 Data Ingestion Layer

The Kafka producer runs as a continuous background process, polling the OpenWeatherMap API on a configurable interval (default 120 seconds). Each API response undergoes enrichment with:

- `location` field for geographic filtering
- `ingestion_time` timestamp for temporal analysis
- `data_type` classification for routing

4.2 Stream Processing Layer

The vector consumer performs three operations for each message:

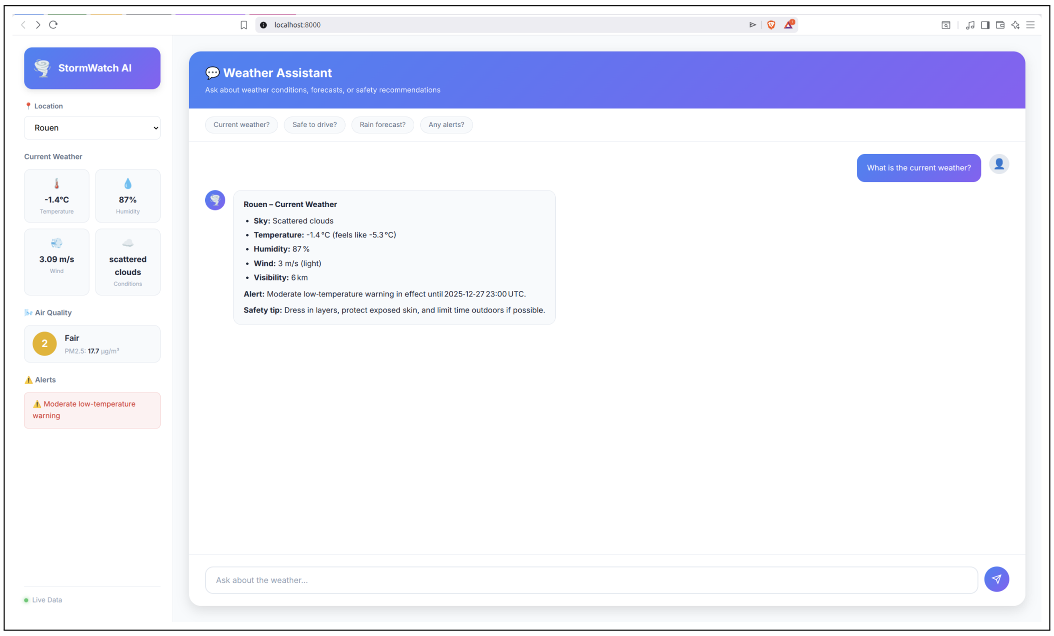
1. **Text Generation** – Construct natural language from structured JSON
2. **Embedding** – Generate 384-dimensional vectors via multilingual sentence transformer
3. **Storage** – Upsert to Qdrant with metadata payload

4.3 News Integration

StormWatch AI ingests French weather news via RSS feeds from Le Monde's climate section, filtering articles by keywords: *vigilance*, *tempête*, *inondation*.

5 Application Interface

The web application provides an intuitive interface for weather queries. Figure 2 shows the main interface.



What is the current weather?

Rouen – Current Weather

- **Sky:** Scattered clouds
- **Temperature:** -1.4 °C (feels like -5.3 °C)
- **Humidity:** 87 %
- **Wind:** 3 m/s (light)
- **Visibility:** 6 km

Alert: Moderate low-temperature warning in effect until 2025-12-27 23:00 UTC.

Safety tip: Dress in layers, protect exposed skin, and limit time outdoors if possible.

Figure 2: StormWatch AI web interface: sidebar with current conditions (left) and chat area for natural language queries (right).

5.1 Interface Components

Table 4: Web interface components

Component	**Features**
Sidebar	City selector, 2×2 metrics grid (temperature, humidity, wind, conditions), alerts section, “Live Data” status badge
Chat Area	Quick-action buttons, custom query input, user/AI message styling with markdown
Header	StormWatch AI branding, assistant description

A critical design decision ensures consistency: all weather data in the sidebar originates from Qdrant, not direct API calls. This ensures that when users ask follow-up questions, the RAG system retrieves the same underlying data they see displayed.

6 Implementation Details

6.1 Embedding Model Selection: Cross-Lingual News Retrieval

The system initially used `all-MiniLM-L6-v2`, an English-optimized sentence transformer. However, StormWatch AI ingests French news from Le Monde RSS feeds, requiring users to query French content with potentially English questions. Since weather data retrieval is deterministic (filter by city/timestamp), only news article retrieval uses semantic RAG, making cross-lingual embedding quality critical.

6.1.1 Benchmark Methodology

We constructed a test set pairing **English user queries** with **French news excerpts** from Le Monde's climate RSS feed. For each pair, we embedded both texts and computed cosine similarity. Table 5 shows the complete test set.

Table 5: Test query-article pairs for cross-lingual news retrieval evaluation

English Query	**French Article Excerpt (Le Monde)**
What are the cold weather protection measures?	Plan Grand Froid : plusieurs préfectures ont actionné des mesures d'urgence pour protéger les plus vulnérables.
What emergency actions have been taken by the government?	Face à la vague hivernale, la Mairie de Paris réclame le déclenchement du plan Grand Froid, l'État répond être largement mobilisé.
Is there a flood warning in the region?	Pluie-inondation : la vigilance orange maintenue dans les Pyrénées-Orientales avec des cumuls de 30 à 50 mm.
Are there any storm warnings?	Pluie-inondations : la vigilance orange levée en Haute-Corse ; les Pyrénées-Orientales maintenues en alerte.
Will it snow tomorrow?	Neige en France : deux départements en vigilance orange, des chutes de 5 à 10 cm localement.
How strong is the wind today?	Vents violents attendus sur la côte atlantique avec des rafales jusqu'à 100 km/h selon Météo-France.

6.1.2 Cross-Lingual Retrieval Results

Table 6: Cross-lingual similarity scores (EN query → FR article)

Query Topic	**MiniLM (EN)**	**Multilingual**	**Improvement**
Cold weather measures	0.18	**0.70**	+288%
Emergency actions	0.21	**0.40**	+92%
Flood warning	0.18	**0.55**	+213%
Storm warnings	0.31	**0.44**	+40%
Snow forecast	0.02	**0.12**	+572%
Wind conditions	0.15	**0.50**	+229%
Average	**0.17**	**0.45**	**+159%**

The multilingual model nearly **triples** cross-lingual similarity scores (0.17 → 0.45). This improvement is critical for news retrieval: scores below 0.3 typically fail to surface relevant articles, while scores above 0.4 reliably match user intent.

6.1.3 Trade-off Analysis

The multilingual model introduces a 27% speed penalty (4,648 vs 6,363 sentences/second). However, both exceed real-time requirements. The cross-lingual gains justify this trade-off for bilingual news retrieval.

Design Decision

StormWatch AI uses `paraphrase-multilingual-MiniLM-L12-v2` to enable cross-lingual queries, accepting a minor speed trade-off for 2× improved retrieval of French weather news.

6.2 Type-Specific Retrieval Strategies

StormWatch AI uses different retrieval strategies depending on data type. Weather data (current conditions, forecasts, air quality) is retrieved **deterministically** using filters and timestamp sorting. Only news articles use **semantic RAG with temporal decay**.

6.2.1 Deterministic Retrieval (Weather Data)

Weather data has a known structure and freshness requirement. Retrieval uses Qdrant's payload filtering without semantic search:

Table 7: Deterministic retrieval methods for structured weather data

Data Type	**Retrieval Method**
Current Weather	Filter by `data_type="current_weather"` and `location`, sort by timestamp descending, return newest entry
Air Quality	Filter by `data_type="air_quality"` and `location`, sort by timestamp descending, return newest entry
Forecast	Filter by `data_type="forecast"` and `location`, sort by timestamp descending, return 2-3 newest entries
Alerts	Filter by `data_type="alert"`, return entries where `start` $\leq$ now $\leq$ `end`

6.2.2 Semantic RAG with Temporal Decay (News Articles)

News retrieval is the only component using true RAG: it combines **semantic similarity** (via embedding cosine distance) with **temporal decay** to prioritize both relevance and freshness.

1. **Embed the user query** using the multilingual sentence transformer
2. **Vector search** in Qdrant filtered by `data_type="news"`
3. **Apply time decay** to re-rank by freshness:

$$s_{\text{final}} = s_{\text{similarity}} \cdot e^{-t/24}$$

where t = article age in hours. Decay rate: 50% retention after 24 hours.

Table 8: News score retention over time

Article Age	Score Retention
1 hour	95.9%
6 hours	77.9%
12 hours	60.7%
24 hours	36.8%

This ensures today's "Plan Grand Froid" article ranks higher than last week's weather news, even if both are semantically similar to the query.

6.3 Location Filtering with Payload Indexes

Qdrant implements efficient filtering via **payload indexes** (B-tree structures on metadata fields):

```
client.create_payload_index(
    collection_name="weather_data",
    field_name="location",
    field_schema=PayloadSchemaType.KEYWORD
)
```

This enables two-phase search: (1) prune to matching payload values, (2) vector similarity within filtered subset. Query latencies remain sub-100ms even as collection grows.

7 Conclusion

This report presented **StormWatch AI**, a real-time weather advisory system that integrates Apache Kafka for stream processing, Qdrant for vector storage, and retrieval-augmented generation with Groq LLM inference.

A central design choice was **differentiated retrieval**: rather than applying uniform RAG to all data types, the system uses database-style filtering for structured weather data, validity-period checking for alerts, and semantic search with temporal decay exclusively for news articles. This approach matches retrieval methods to data characteristics, ensuring both accuracy and efficiency.

The architecture combining Kafka, vector databases, and RAG represents a reusable pattern for domains requiring conversational access to streaming data, such as financial news analysis, IoT sensor monitoring, and real-time event tracking.

A potential extension would be to leverage LLM tool-use capabilities, where the model dynamically generates Qdrant queries based on user intent. This would increase flexibility but adds latency and complexity; our explicit retrieval logic prioritizes reliability and speed for the well-defined weather domain.

References

[1] Kreps, J., Narkhede, N., and Rao, J. (2011). Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB Workshop at SIGMOD*.

[2] Qdrant Team. (2024). Qdrant: Vector similarity search engine with extended filtering support. `https://qdrant.tech/`

[3] Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.

[4] OpenWeatherMap. (2024). Weather API documentation. `https://openweathermap.org/api`

[5] Groq Inc. (2024). Groq LPU Inference Engine. `https://groq.com/`

[6] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of EMNLP 2019*, 3982–3992.