

Final Presentation – Real-Time Big Data Processing 2024/2025

# Real-Time Analysis of Public GitHub Activity

Roland Bernard

February 2026

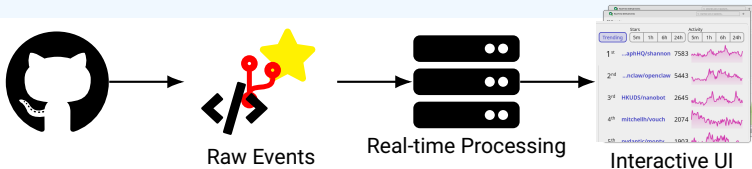
# Application Domain

- Motivation and Context:

- GitHub hosts millions of developers and repositories.
- Generates large volumes of public events.
- Most existing tools focus mainly on historical analysis.

- Project Objectives:

- Collect GitHub events continuously.
- Process data in real time.
- Provide interactive dashboard.
- Ensure low latency and reliability.



# Description of the Data Sources

## ▶ GitHub Events API:

- Endpoint: <https://api.github.com/events>.
- Up to 300 most recent public events in JSON format.
- Rate limited, i.e., access token required.

## ▶ Key Properties:

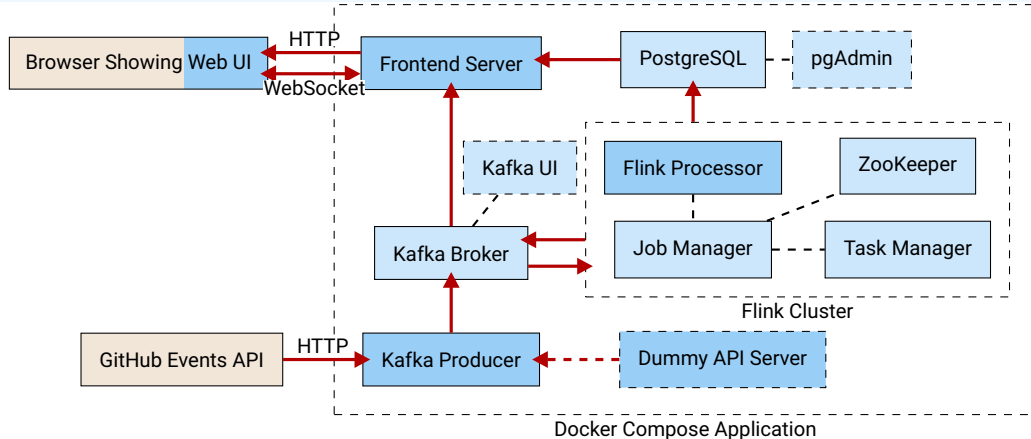
- Variety: many event types, e.g., push, issues, stars, etc.
- Velocity: up to 120 events/sec, millions of events per day.
- Veracity: out-of-order and delayed events.
- Latency: API around 5 minutes behind real time.

```
{
  "id": "8251545586",
  "type": "PushEvent",
  "actor": {
    "id": 121951544,
    "login": "movieflixgr",
    "display_login": "movieflixgr",
    "gravatar_id": "",
    "url": "https://api.github.com/users/[",
    "avatar_url": "https://avatars.github[",
  },
  "repo": {
    "id": 1126450259,
    "name": "movieflixgr/Subtitles",
    "url": "https://api.github.com/repos/[",
  },
  "payload": {
    "repository_id": 1126450259,
    "push_id": 30546526697,
    "ref": "refs/heads/main",
    "head": "d5e986993aa47729cb18a82ac9d1[",
    "before": "6bf91f2d0600084cc7218c1934[",
  },
  "public": true,
  "created_at": "2026-02-
08T20:09:50Z"
}
```

# System Architecture

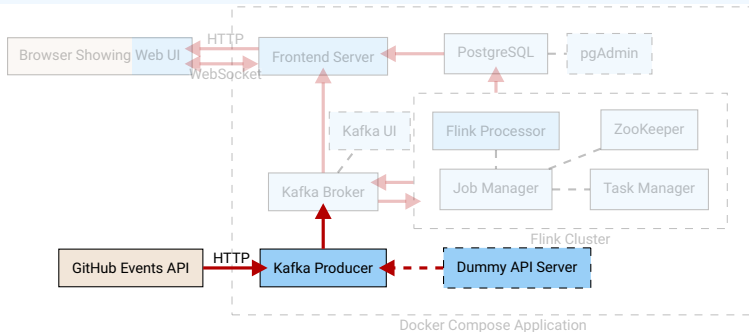
## ► Technology Stack:

- Backend: Java + Maven; RxJava; Apache Kafka; Apache Flink; PostgreSQL + TimescaleDB
- Frontend: Single Page Application with React + TypeScript; Vite; Recharts; RxJS
- Deployment: Docker and Docker Compose



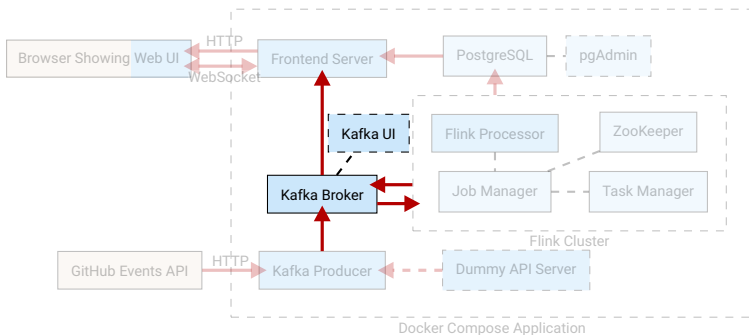
# System Architecture – Data Ingestion

- ▶ **Kafka Producer:**
  - Polls GitHub API periodically, respecting rate limits.
  - Retrieves overlapping windows, then deduplicates events.
  - Publishes deduplicated events to Kafka topic.
- ▶ **Testing Mode:**
  - Dummy API server to simulate real GitHub API with configurable throughput.



# System Architecture – Messaging

- ▶ Apache Kafka:
  - Decouples the ingestion from the processing.
  - Decouples the processing from the frontend.
  - Kafka message retention enables restarting Flink job without losing events.



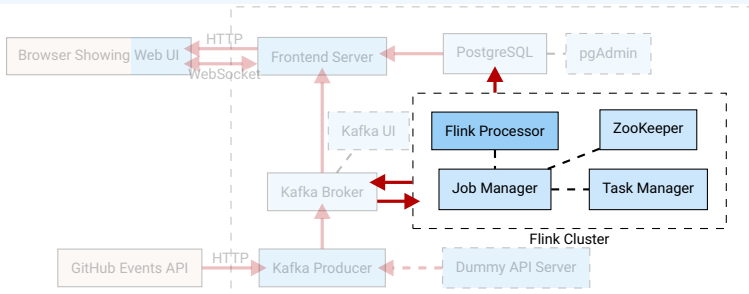
# System Architecture – Stream Processing

- Core Processing Tasks:

- Retrieve raw events from Kafka and parse JSON.
- Event-time processing using Watermarks with 10s maximum out-of-orderness.
- Writes results to PostgreSQL and Kafka.

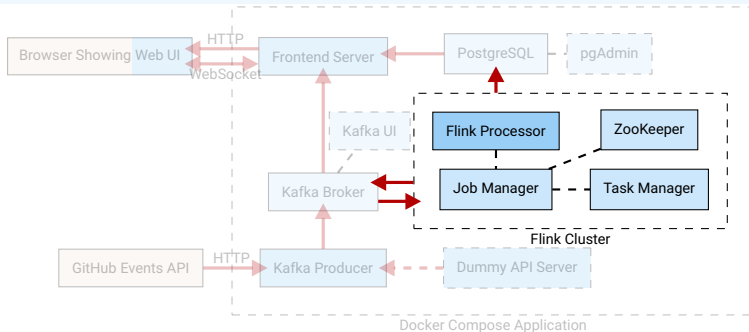
- Windowed Aggregations:

- Tumbling windows with 10s and 5min window sizes.
- Custom sliding windows with 5min, 1h, 6h, and 24h length, updated every second.



# System Architecture – Stream Processing

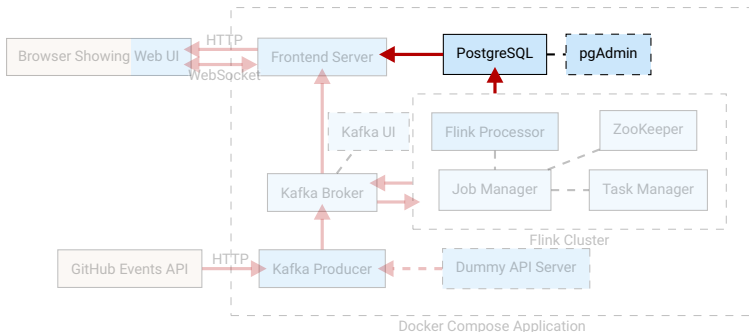
- ▶ Leaderboards:
  - User and repository activity ranking.
  - Based on latest values from sliding window aggregations.
  - Optimization: Emits only changed rows and frontend reconstructs ranking.
- ▶ Trending Score:
  - Weighted sum of recent stars:  $t_{\text{score}} = 10s_{5m} + 5s_{1h} + 2s_{6h} + s_{24h}$ .





# System Architecture – Storage

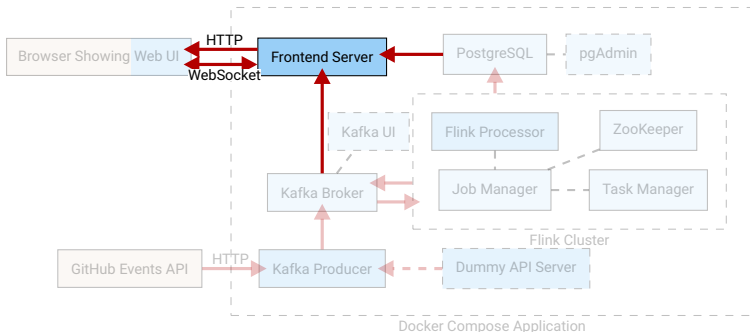
- ▶ PostgreSQL + TimescaleDB:
  - Persistent storage for results of Flink processor.
  - Handles queries for initial snapshot and historical data.
  - Makes use of TimescaleDB hypertables and retention policy.
  - Database commit before Kafka publish to ensure consistency.



# System Architecture – Data Serving

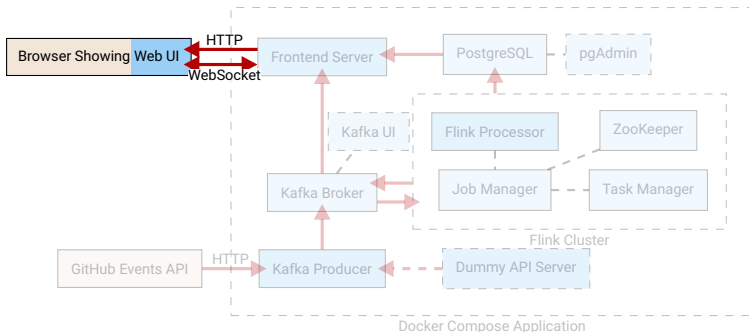
## ▸ Frontend Server:

- HTTP server for serving static files of the client web application.
- WebSocket server to serve API.
- Combines database queries for snapshots and Kafka streams for real-time updates.
- Manages client subscriptions to route events received from Kafka.



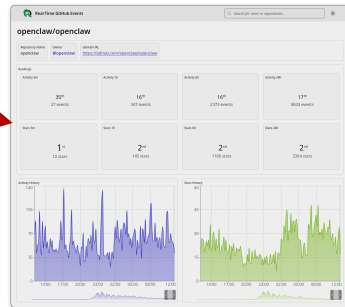
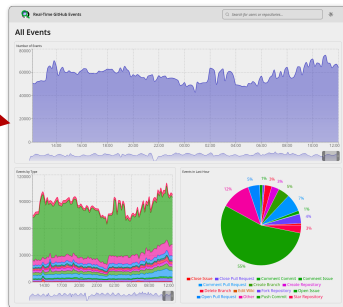
# System Architecture – Frontend

- ▶ Web Application:
  - A filterable live event stream.
  - WebSocket server to serve API.
  - Real-time counters for event volumes.
  - User and repository leaderboards.



# Functionalities and Demo

- ▶ Live Demo: <https://rtgh.rolandb.com>



# Conclusion

## ▶ Successes:

- Robust architecture able to handle 70-120 events per second.
- Docker and Docker Compose simplifies deployment.
- Responsive UI with leaderboards and charts updated in real-time.

## ▶ Challenges:

- Real-time rankings and long sliding window efficiency.
- API changes made by GitHub reduced information.
- Debugging Flink operator logic and state management.

## ▶ Future Improvements:

- More advanced models to detect trending repositories.
- Social media integration to see where repositories or users are mentioned.
- Sentiment analysis on the text content of certain event types.

