Real-Time Wikimedia Streaming Analytics

📌Overview

This project is an **end-to-end real-time streaming analytics platform** that ingests live Wikipedia edit events, processes them in real time, and stores aggregated metrics for low-latency analytical querying.

The system is designed to mirror **production-grade streaming architectures** used at companies like TikTok, Netflix, and Uber.

💣Architecture

```
Wikimedia EventStreams (Live SSE API)
        ↓
Kafka (Real-time Event Log)
        ↓
Apache Flink (PyFlink Stream Processing)
        ↓
ClickHouse (Real-time Analytics Database)
        ↓
SQL / Dashboards
```

⚙️Technologies Used

- **Apache Kafka** – real-time data ingestion and buffering
- **Apache Flink (PyFlink)** – event-time stream processing
- **ClickHouse** – low-latency analytical database
- **Docker & Docker Compose** – containerized deployment
- **Python** – streaming logic
- **SQL** – analytics and validation

🕐Data Flow

✂️Data Ingestion

- Live Wikipedia edit events are streamed from **Wikimedia EventStreams**
- Events include wiki name, timestamp, and bot flag
- Events are continuously published into a Kafka topic (`wikimedia_events`)

✒️Kafka

- Acts as a durable, scalable event log
- Decouples producers and consumers
- Enables fault tolerance and replayability

🛏️Apache Flink (PyFlink)

- Consumes events from Kafka
- Applies **event-time processing and watermarks**
- Filters bot edits
- Aggregates edit counts using **1-minute tumbling windows**
- Processes data continuously in memory

🌂ClickHouse

- Stores aggregated results via JDBC sink
- Enables millisecond-level analytical SQL queries
- Serves as the real-time analytics store

🗃️ClickHouse Schema

```
CREATE TABLE analytics.wiki_edits (
    wiki String,
    window_start DateTime,
    edit_count UInt64
)
ENGINE = MergeTree()
ORDER BY (wiki, window_start);
```

🗜️How to Run

✂️Start All Services

```
docker-compose up -d
```

Services started: - Kafka - Zookeeper - Flink JobManager & TaskManager - ClickHouse

---

✒️Run PyFlink Job

```
docker exec -it flink-jobmanager
/opt/flink/bin/flink run -py /opt/flink/wiki_stream.py
```

Check Flink UI:

```
http://localhost:8081
```

🛏️Verify Data in ClickHouse

```
docker exec -it clickhouse clickhouse-client
```

```
SELECT *
FROM analytics.wiki_edits
ORDER BY window_start DESC
LIMIT 10;
```

New rows appear **every minute**.

📊Sample Analytics Queries

📀Top Wikis (Last 5 Minutes)

```
SELECT wiki, SUM(edit_count) AS total_edits
FROM analytics.wiki_edits
WHERE window_start > now() - INTERVAL 5 MINUTE
GROUP BY wiki
ORDER BY total_edits DESC;
```

📈Edit Volume Over Time

```
SELECT window_start, SUM(edit_count)
FROM analytics.wiki_edits
GROUP BY window_start
ORDER BY window_start;
```

🧪Validation & Testing

- Verified Kafka topic creation and consumption
- Confirmed Flink job execution via Flink UI
- Manually tested ClickHouse inserts
- Validated real-time data ingestion using SQL queries

Key Engineering Concepts Demonstrated

- Real-time streaming architectures
- Kafka topic and broker management
- Event-time vs processing-time semantics
- Windowed aggregations
- Stream-to-analytics database integration
- Docker-based deployment

🕥Why This Project Is Strong

- Uses **production-grade technologies**
- Demonstrates **real-time data engineering skills**
- End-to-end pipeline with validation
- SQL-accessible real-time analytics

📄Resume Summary

Built a real-time streaming analytics platform using Kafka, PyFlink, and ClickHouse to process live Wikimedia events with event-time windowing and low-latency analytical queries.

🔮Future Enhancements

- Grafana dashboards on ClickHouse
- Exactly-once processing with Flink checkpoints
- Alerting on edit spikes
- Flink SQL implementation
- Historical analysis with Hive / Presto

Author

Lohit

⭐If you find this project useful, feel free to star the repository!