PulseSky: Trend Intelligence System

"Taking the pulse of decentralized social conversations."

Team SOBE

Team Member	Primary Role	Specialization
Olga Grigorieva	Lead Data Engineer	Cloud Architect, OpenSearch, Grafana, Metastore
Elen Muradyan	Data Engineer	NiFi Flows, Superset Dashboards, Visualization
Suren Mnatsakanyan	Data Engineer	Kafka Cluster, Data Enrichment, Low-latency Streams
Bartosz Maj	Data Engineer	Spark Streaming, HDFS Architecture, Batch ML Pipelines

1. High-Level Project Description

1.1 Overview

PulseSky is a Big Data analytics platform designed to extract, process, and visualize emerging trends and sentiments from decentralized social networks, starting with **Bluesky** and **Twitter (X)**.

The system captures the pulse of public conversation in real time, enabling marketers, analysts, and researchers to identify rising topics, measure sentiment, and map influential voices within open digital ecosystems.

1.2 User Perspective

From the user's perspective:

- **Strategic analysts** can monitor live discussions about brands or competitors.
- **Researchers** can study how sentiment evolves across decentralized communities.
- **Startup managers** can identify early interest in new technologies or categories.

The user interface consists of an interactive dashboard featuring:

- Live top-trending topics and sentiment graphs.
- Language- and region-based filters.
- Influencer leaderboards and engagement analytics.

1.3 Core Use-Case

"A marketer wants to know what topics are trending in the sustainability and tech communities on Bluesky and Twitter to plan upcoming campaigns."

The system ingests live posts, performs NLP-based topic modeling and sentiment analysis, and presents insights such as:

- Trending keywords and hashtags.
- Positive/negative sentiment ratio.
- Trend growth and momentum analysis.

1.4 Business Process Diagram

The high-level business process of **PulseSky** is shown in Figure . It illustrates how decentralized social data from Bluesky and Twitter is ingested, processed, stored, and visualized for end users such as marketers, researchers, and startup managers.

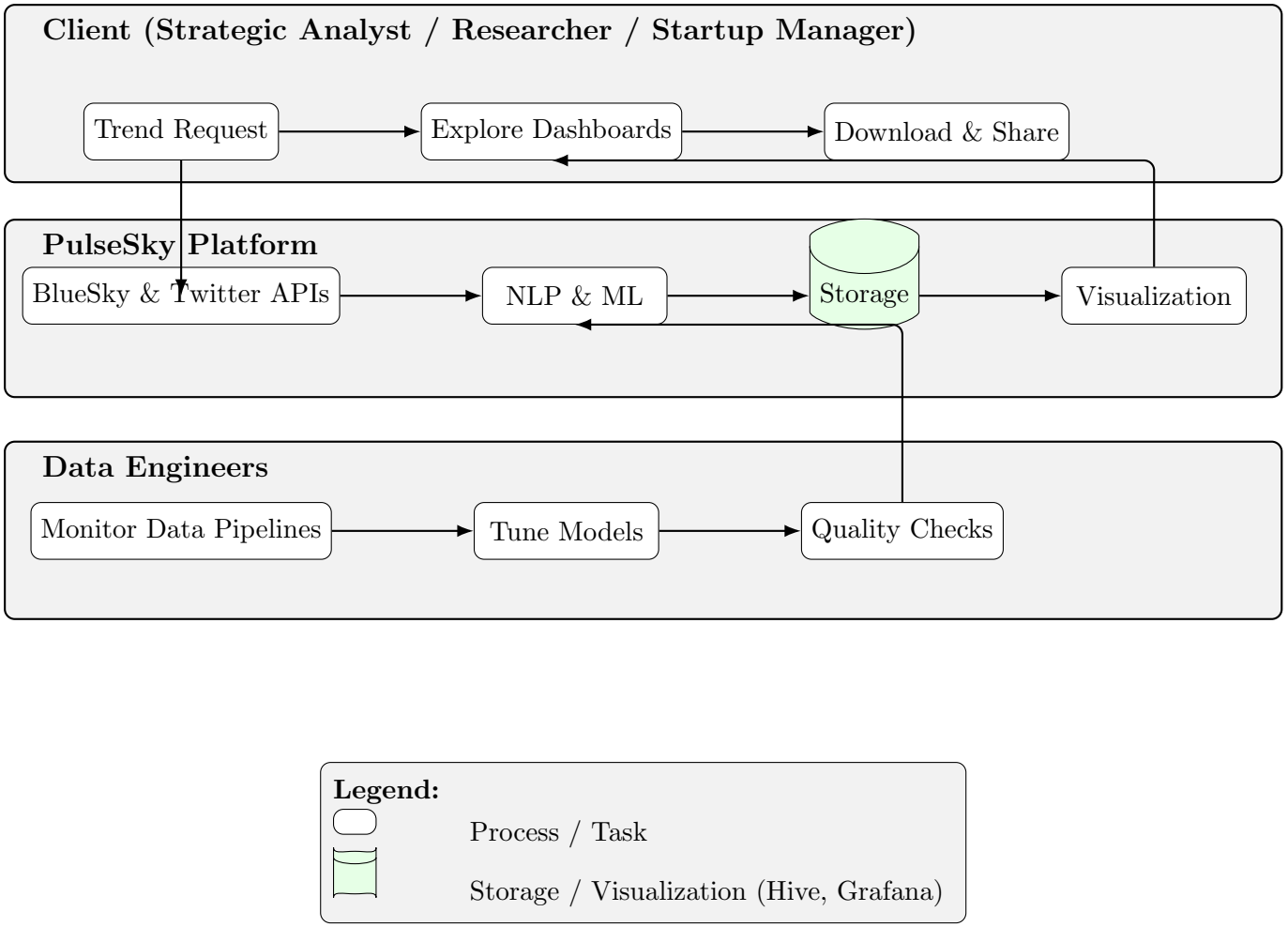

Figure 1.1: PulseSky: High-Level Business Process Overview

2. Preliminary Data Investigation

This chapter provides an overview of the data landscape underpinning the PulseSky Trend Intelligence System. It outlines the primary and supplementary data sources, the key characteristics of each dataset, and preliminary analytical insights that inform the project's design and processing assumptions.

2.1 Data Sources

The PulseSky platform integrates decentralized and centralized social media streams to enable comprehensive sentiment and trend analysis.

- **Primary Source:** *Bluesky AT Protocol Firehose API* - an open, federated data stream providing real-time access to public conversations across decentralized communities. It supports rich metadata such as content type, hashtags, timestamps, and author identifiers.
- **Supplementary Source:** *Twitter (X) API* - used to complement the decentralized data with centralized social discourse, allowing comparative trend and sentiment studies between platforms.

2.2 Data Characteristics

The data collected from these sources exhibits the typical traits of high-volume social media streams. Table 2.1 summarizes the core characteristics.

Characteristic	Description
Type	Text-based posts containing user-generated content, hashtags, metadata, timestamps, and engagement statistics.
Format	JSON structures defined by the AT Protocol schemas for Bluesky; REST-based JSON responses from Twitter (X).
Volume	Approximately 1–5 million posts per day (depending on filters and languages) for Bluesky; 1000 requests/sec via the third-party Twitter API.
Velocity	Real-time streaming ingestion from Bluesky through a WebSocket Firehose, coupled with daily batch aggregation; websocket ingestion coupled with daily batch aggregations for Twitter API data.

Variety	Multiple post types including original posts and replies
Veracity	Public, user-generated content with inherent variability in accuracy, bias, and completeness; mitigated through deduplication and cleaning steps.

Table 2.1: Summary of Data Characteristics for Primary and Supplementary Sources

2.3 Preliminary Data Insights

Initial exploratory analyses of sample data from the Bluesky Firehose stream highlight a diverse conversational landscape with clusters forming around major thematic areas such as technology, sustainability, culture, and policy. This natural topical segmentation supports PulseSky's objective of identifying early signals of emerging trends and community sentiment.

Further inspection of the data revealed the following observations:

- **Conversation Structure:** Bluesky data includes clear linkage between posts and their replies, enabling community-level sentiment tracking and conversational context extraction.
- **Cross-Platform Signals:** Integrating Bluesky and Twitter allows comparison between decentralized and centralized discourse ecosystems, providing richer trend validation.
- **Data Integrity:** While Twitter (X) data may include edit histories (for premium users), these are considered non-essential for trend analytics and excluded from ingestion.
- **Analytical Suitability:** The mixture of textual, temporal, and interaction data provides a strong foundation for topic modeling, sentiment analysis, and influencer detection.

2.4 Preliminary Conclusion

The preliminary investigation confirms that the data sources chosen - primarily Bluesky, supplemented by Twitter (X) - are well-suited to the PulseSky system's mission of tracking decentralized social sentiment. Their real-time nature, volume, and variety make them ideal for building an automated pipeline capable of surfacing emerging trends in near real time.

3. Key Assumptions and Data Processing Flow

This chapter documents the key assumptions and data processing logic behind the PulseSky pipeline. It explains how data flows through ingestion, enrichment, processing, and analytics layers - from decentralized social platforms to actionable insights - as illustrated in Figure 3.1.

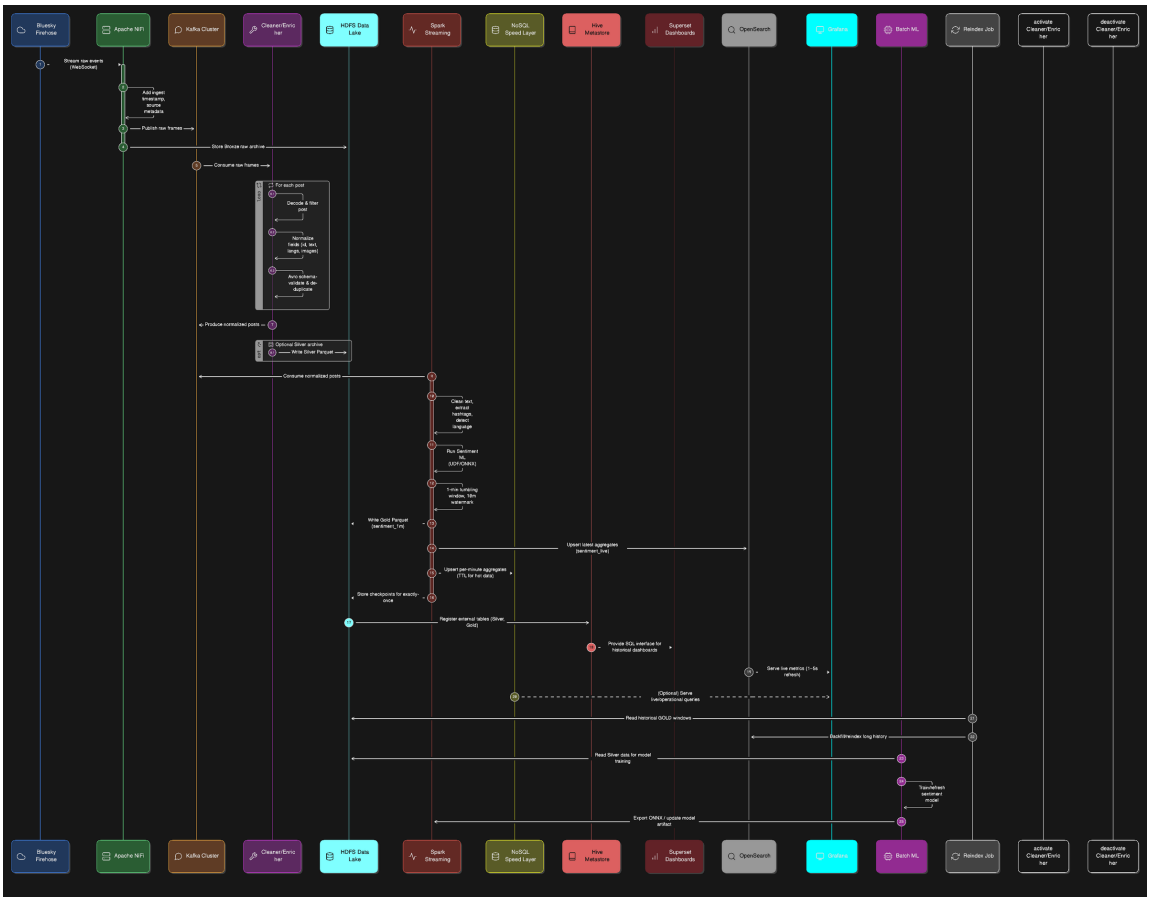

Figure 3.1: PulseSky High-Level Data Flow Diagram

3.1 Overview of Processing Assumptions

The PulseSky system operates under the following key assumptions for both real-time streaming and batch analytics:

- **Schema Consistency:** All records adhere to Avro schemas validated at ingestion

and enrichment stages.

- **Event-Time Semantics:** Spark Streaming uses event timestamps with 10-minute watermarks to manage late-arriving data.
- **Deduplication:** Cleaner/Enricher and Spark apply ID-based de-duplication to prevent record duplication.
- **Multi-tier Storage:** The data lake follows a Bronze–Silver–Gold structure for progressive data refinement.
- **Low-Latency Analytics:** Spark and OpenSearch provide live analytical views updated within seconds.

3.2 Processing Layers

Apache NiFi

- Acts as a reliable WebSocket–Kafka bridge.
- Consumes Firehose data, adds `ingest_ts` and provenance metadata.
- Optionally stores raw frames in the Bronze layer of HDFS.
- Publishes records to Kafka for downstream enrichment.

Cleaner / Enricher (Java)

- Decodes Firehose frames into post-level records.
- Normalizes, validates, and de-duplicates by post ID.
- Outputs `posts_clean` topic (Avro) to Kafka.
- Optionally writes normalized posts to Silver (Parquet) on HDFS.

Apache Spark (Python)

- Reads `posts_clean` from Kafka using Structured Streaming.
- Cleans text, extracts topics, and computes sentiment via Spark NLP.
- Uses 1-minute tumbling windows with 10-minute watermarks.
- Writes results to:
  - Gold Parquet (HDFS) for historical analytics.
  - OpenSearch index (`sentiment_live`) for real-time dashboards.

Hive + Superset

- Hive exposes Parquet data (Silver & Gold) as external tables.
- Superset connects to Hive for interactive dashboards.
- Dashboards auto-refresh every 30–60 seconds for analytical exploration.

OpenSearch + Grafana

- Stores near-real-time sentiment aggregates in `sentiment_live`.
- Grafana and Kibana provide 1–5 second refresh dashboards.
- Enables leaderboards, counters, and time-based sentiment tracking.
- Uses NoSQL for near real-time search and analytics,

3.3 Data Lake Layout (HDFS)

- **Bronze (Raw):** `hdfs:///lake/bronze/firehose_raw/ingest_date=YYYY-MM-DD/hour=HH/`
- **Silver (Clean):** `hdfs:///lake/silver/posts/date=YYYY-MM-DD/hour=HH/lang=XX/`
- **Gold (KPIs):** `hdfs:///lake/gold/sentiment_1m/date=YYYY-MM-DD/`

3.4 High-Level Data Flow Summary Diagram

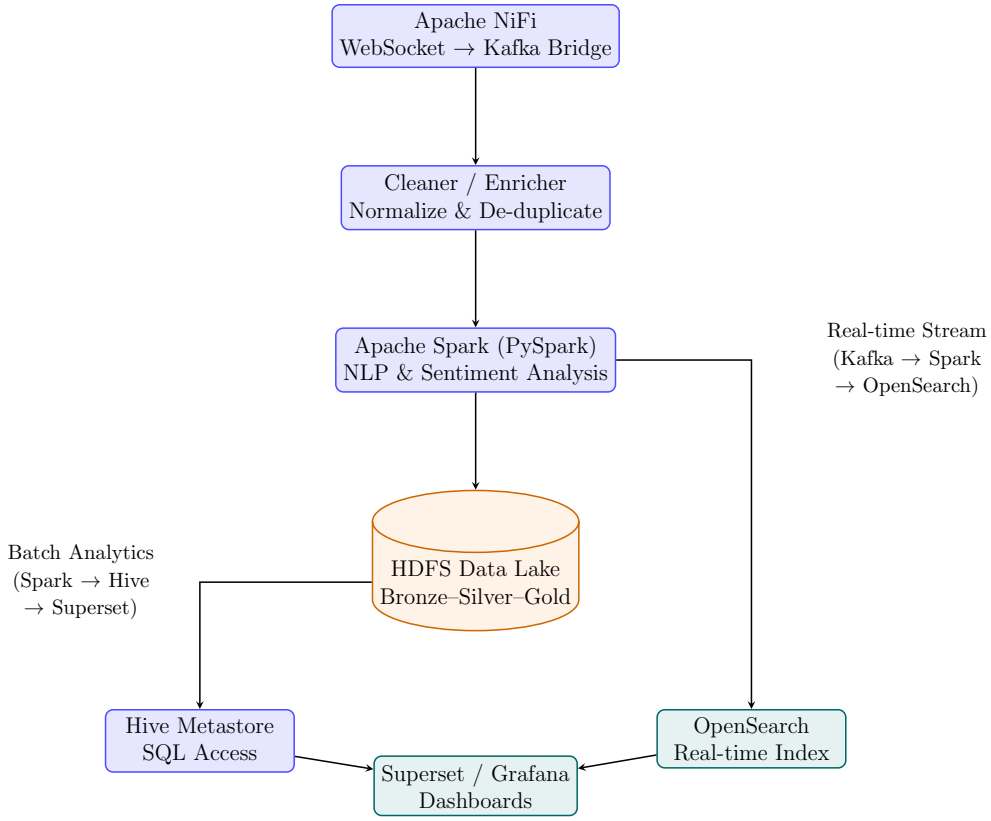

Figure 3.2: Simplified Technical Data Flow of the PulseSky Processing Pipeline

3.5 Summary

The PulseSky data flow integrates streaming and batch analytics to deliver both immediate and historical insights. Each component contributes to a unified architecture - NiFi for ingestion, Kafka for transport, Spark for computation, Hive for query abstraction, and OpenSearch/Superset for visualization - ensuring the system continuously "takes the pulse" of decentralized social conversations.

4. Solution Infrastructure

This chapter outlines the end-to-end technical architecture of the PulseSky Trend Intelligence System, including the Big Data technologies, platforms, and tools used across each layer of the data pipeline. The design emphasizes scalability, modularity, and real-time responsiveness - enabling the system to ingest decentralized social data streams, process them efficiently, and deliver actionable insights via interactive dashboards.

4.1 Ingestion Layer

Apache NiFi

Apache NiFi acts as a reliable bridge between the decentralized data sources and the central processing system. It manages the flow of live post events from the **Bluesky Firehose API** and scheduled ingestions from the **Twitter (X) API**.

- Establishes a WebSocket connection to the Bluesky Firehose for real-time post capture.
- Adds ingestion timestamps, provenance metadata, and unique identifiers for traceability.
- Routes data through custom processors before publishing to Kafka topics.

Apache Kafka

Kafka serves as the streaming backbone of the PulseSky platform, ensuring reliable, high-throughput transport of social data between components.

- Utilizes Avro serialization and schema validation to maintain message consistency.
- Provides decoupled communication between ingestion (NiFi) and analytics (Spark) layers.
- Implements two main topics:
  - **posts_raw** – unprocessed messages directly from NiFi.
  - **posts_clean** – normalized, de-duplicated messages enriched with language and source metadata.

4.2 Data Storage Layer

HDFS Data Lake

The Hadoop Distributed File System (HDFS) serves as the central data repository, supporting both raw and processed data storage in structured zones:

- **Bronze Zone:** Stores raw Firehose frames from the Bluesky API.
- **Silver Zone:** Holds cleaned and normalized post data in Parquet format.
- **Gold Zone:** Contains 1-minute aggregated sentiment KPIs for trend analytics.

Hive Metastore

The Hive Metastore provides schema abstraction over the data lake, allowing SQL-based querying of Parquet data.

- Hosts external tables such as `silver.posts` and `gold.sentiment_1m`.
- Enables business intelligence tools like Superset to directly query analytical datasets.

AWS S3 (Optional Extension)

AWS S3 is used for off-site backup, long-term storage, and archiving of Parquet datasets or ML artifacts. This ensures redundancy and facilitates model versioning and historical trend analysis.

4.3 Data Processing & Analytics Layer

Apache Spark (PySpark)

Spark provides the distributed data processing backbone for both streaming and batch workloads.

- Cleans and tokenizes text, extracts topics, and computes sentiment scores using Spark NLP.
- Processes streaming data with 1-minute tumbling windows and event-time watermarks.
- Writes aggregated outputs simultaneously to:
  - The **Gold Parquet Layer** (HDFS) for historical analytics.
  - The **OpenSearch sentiment_live index** for real-time dashboards.

4.4 Modeling & Machine Learning Layer

Python (PyTorch / Transformers)

The ML layer uses pretrained and fine-tuned transformer models to detect sentiment and contextual meaning within social posts.

- Integrates models such as **BERT** and **RoBERTa** via Spark NLP for distributed inference.
- Supports both streaming inference (live dashboards) and batch scoring (historical trend modeling).
- Continuously improves sentiment classification through periodic fine-tuning on social text datasets.

4.5 Analytics, Visualization & Monitoring Layer

OpenSearch + Kibana / Grafana

OpenSearch indexes real-time outputs from Spark Streaming for low-latency exploration and visualization.

- Stores near-real-time sentiment data in the `sentiment_live` index with second-level latency.
- Kibana and Grafana dashboards display:
  - Topic leaderboards.
  - Sentiment distributions by language and time.
  - Live counters and trend evolution charts.
- Enables 1–5 second auto-refresh for continuous insight updates.

Apache Superset

Superset connects to the Hive Metastore for analytical dashboards and data exploration.

- Visualizes aggregated KPIs (e.g., average sentiment, post volume, topic growth).
- Supports flexible slicing and filtering by date, topic, and language.
- Refresh interval configured for 30–60 seconds to balance freshness with performance.

Grafana (Monitoring)

The monitoring stack ensures observability and operational reliability across all system layers.

- Grafana visualizes system health indicators including:
  - Kafka lag and broker uptime.
  - Spark job latency and error rates.
  - NiFi throughput and queue utilization.
- Alerts notify the engineering team of SLA breaches or component failures.

4.6 Infrastructure Summary

PulseSky's modular infrastructure enables continuous ingestion, near-real-time analytics, and high-availability insights. By combining open-source Big Data frameworks (NiFi, Kafka, Spark, Hive) with advanced ML and visualization tools (OpenSearch, Superset, Grafana), the platform delivers both scalability and transparency - empowering researchers and marketers to take the pulse of decentralized social conversations.

5. Team Members and Responsibilities

The PulseSky project is driven by a multidisciplinary team combining expertise in cloud architecture, data engineering, stream processing, and data visualization. Each team member contributes to a distinct layer of the system - from decentralized data ingestion to machine learning analytics and trend visualization - ensuring an integrated, reliable, and scalable solution.

Olga Grigorieva - Data Engineer & Project Lead

Olga provides overall direction and coordination across all project components and ensures that data flows seamlessly through the pipeline and that the system infrastructure aligns with scalability, reliability, and analytical needs.

Key Responsibilities:

- Lead project planning, team coordination, and deliverable tracking.
- Architect the cloud infrastructure and deployment environment.
- Manage the Hive Metastore to maintain structured schemas for Silver and Gold datasets.
- Integrate and configure **OpenSearch** for real-time analytics.
- Develop **Grafana** dashboards for monitoring and insights.
- Oversee technical documentation, version control, and PR reviews.

Suitable Replacement: Elen Muradyan

Elen Muradyan - Data Engineer & Visualization Specialist

Elen is responsible for ingesting decentralized social data streams and transforming them into analytics-ready formats. Also bridges the ingestion and visualization layers, ensuring data is accurately represented in dashboards.

Key Responsibilities:

- Implement **Apache NiFi** flows to capture data from the Bluesky Firehose and Twitter APIs.

- Configure NiFi-to-Kafka pipelines for high-throughput, fault-tolerant ingestion.
- Define transformation logic for structured topic and sentiment fields.
- Build and maintain **Apache Superset** dashboards for trend monitoring.
- Design visual metrics and KPIs trending topics, sentiment ratios, and influencer reach.
- Collaborate with the Spark team to align schema structures and analytical outputs.

Suitable Replacement: Olga Grigorieva

Suren Mnatsakanyan - Data Engineer

Suren designs and maintains the backbone of PulseSky's real-time data streaming architecture. His work ensures that the ingestion, cleaning, and enrichment processes are optimized for latency and data quality.

Key Responsibilities:

- Deploy and manage the **Kafka Cluster** for distributed message streaming.
- Develop the **Cleaner/Enricher** service in Java to normalize and validate post data.
- Implement Avro schemas for message serialization and schema validation.
- Define Kafka topic hierarchy for raw, cleaned, and analytical datasets.
- Optimize delivery between **NiFi**, **Kafka**, and **Spark** for real-time streaming.
- Support model integration by preparing input streams for ML inference.

Suitable Replacement: Bartosz Maj

Bartosz Maj - Data Engineer & ML Pipeline Developer

Bartosz is responsible for the processing and analytical intelligence layers of PulseSky. He builds and optimizes the Spark-based pipelines that transform social data into meaningful insights.

Key Responsibilities:

- Design and maintain the **HDFS** data lake (Bronze, Silver, Gold zones).
- Develop **Spark Streaming** jobs for topic modeling and sentiment aggregation.
- Implement batch pipelines for trend forecasting and cross-lingual sentiment tracking.

- Integrate **Spark NLP** models (BERT, RoBERTa) for contextual sentiment analysis.
- Automate ETL workflows and scheduling with **Apache Airflow**.
- Ensure consistency between the analytical datasets and Superset/OpenSearch outputs.

Suitable Replacement: Suren Mnatsakanyan

Team Collaboration and Workflow

The PulseSky team follows agile and data-driven collaboration practices to ensure continuous progress and integration across layers.

Collaboration Practices:

- Weekly meetings focused on incremental feature delivery and cross-component testing.
- Shared GitHub repository for version control, with submodules, feature branching and PR review cycles.
- Centralized ReadMe file for documentation, architecture diagrams, and API references.
- Regular design syncs between ingestion (NiFi/Kafka), processing (Spark), and visualization (Superset) teams.