Project Report: Real-Time FinTech Market Sentiment Analysis Pipeline

Author: Prabudh Raghuram

1. Summary

This report details the design, implementation, and outcomes of a real-time data engineering project developed to analyse and visualize sentiment in the FinTech market. The primary objective was to build a robust, end-to-end data pipeline using modern cloud technologies to demonstrate skills relevant to the UK job market in data engineering, business analytics, and FinTech.

The project successfully ingests a continuous stream of synthetic FinTech news headlines generated by a local Python script. This data is streamed into **Microsoft Azure**, where it is processed in real-time by **Azure Stream Analytics**. The processed data, enriched with sentiment scores, is then stored in an **Azure SQL Database**. Finally, a live, interactive dashboard built in **Google Looker Studio** connects to this database, providing an up-to-the-minute view of market sentiment, company-specific trends, and headline volume.

The project effectively demonstrates proficiency in cloud architecture, real-time data processing, database management, and business intelligence. Key challenges, including cloud service integration and data pipeline debugging, were systematically resolved, highlighting practical problem-solving abilities. The result is a live dashboard that translates raw data into actionable business insights.

2. Introduction

2.1. Problem Statement

In the fast-paced Financial Technology (FinTech) sector, market sentiment can shift in minutes, driven by news, regulatory announcements, and technological breakthroughs. Stakeholders, from investors to product managers, require immediate insights into market trends to make timely, informed decisions. Traditional batch-processing analytics systems often have a significant time lag, rendering their insights obsolete by the time they are delivered. There is a critical need for a solution that can ingest, process, and visualize market data in near real-time.

2.2. Project Objectives

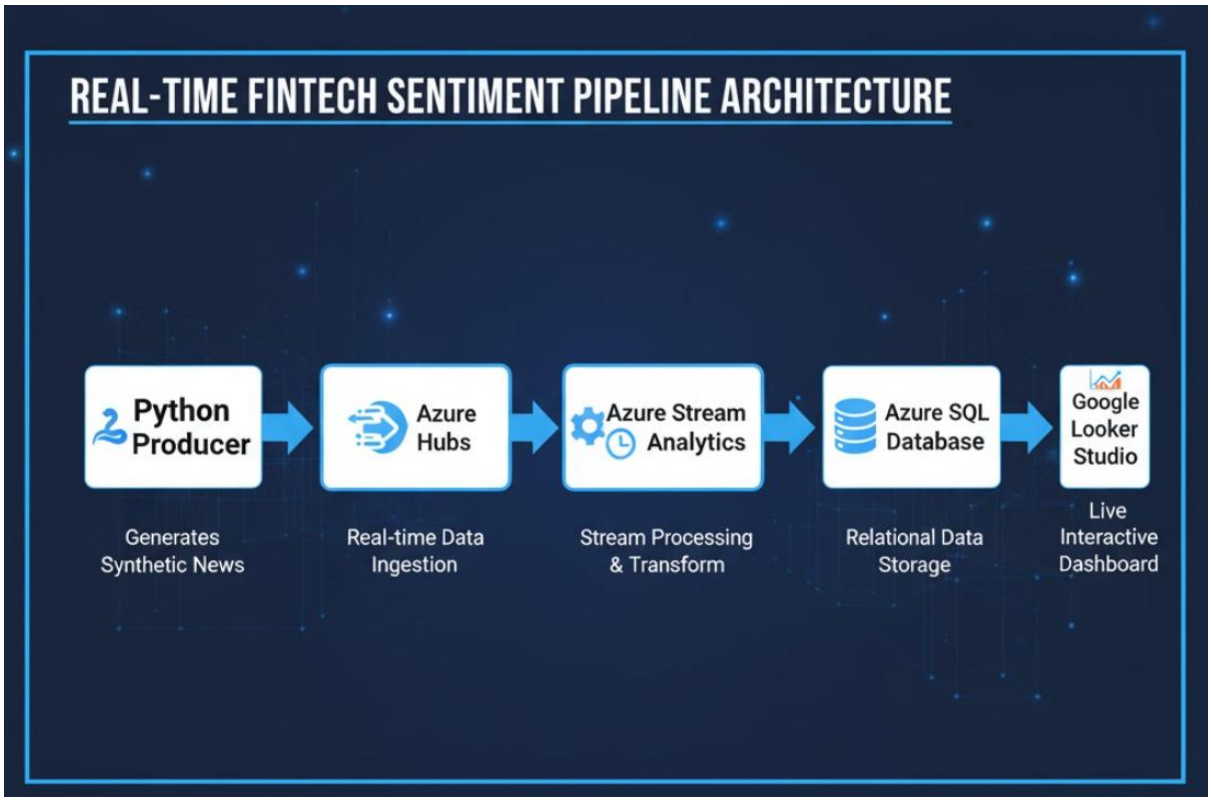
The project was designed to address this problem statement by achieving the following objectives:

1. **Design and build an end-to-end data pipeline** capable of handling a continuous stream of data.
2. **Utilize a modern, cloud-native stack** centred on Microsoft Azure to demonstrate proficiency in industry-standard tools.

3. **Perform real-time data transformation**, including natural language processing (NLP) for sentiment analysis.
4. **Store the processed data** in a scalable and reliable cloud database.
5. **Develop a live, interactive dashboard** to visualize key performance indicators (KPIs) and provide actionable insights.
6. **Document the entire process**, including challenges and architectural decisions, to create a comprehensive portfolio piece.

3. Technical Architecture & Design

The project's architecture is designed for scalability, reliability, and real-time performance. It comprises five distinct stages, from data generation to visualization.



3.1. Phase 1: Data Production & Ingestion

- **Component: Python Producer Script**

  A local Python script was developed to act as the data source. It uses the Faker library to generate realistic, synthetic FinTech company names and news events.

  The **NLTK (VADER)** library is used for on-the-fly sentiment analysis, classifying each headline as Positive, Negative, or Neutral and assigning a compound sentiment score.

  This component simulates a real-world stream of unstructured text data.

- **Component: Azure Event Hubs**

  The Python script sends the generated JSON data payloads to an Azure Event Hub (eh-news-headlines).

  Event Hubs acts as the highly scalable, high-throughput ingestion point for our data pipeline, capable of handling millions of events per second. It decouples the data producer from the data processor.

3.2. Phase 2: Real-Time Processing

- **Component: Azure Stream Analytics (ASA)**

  An ASA job (asa-sentiment-processor-uk) is configured to read data directly from the Event Hub input stream.

  It runs a continuous, SQL-like query to select and structure the incoming JSON data, preparing it for storage. The query used was:

  SELECT

  System.Timestamp AS EventTimestamp,headline,company,sentiment,compound_score

  INTO

  [sqldb-output]

  FROM

  [eventhub-input]

3.3. Phase 3: Data Storage

- **Component: Azure SQL Database**

  The output of the Stream Analytics job is written directly into an Azure SQL Database (db-fintech-sentiment).

  This serves as the persistent, relational data store for our processed, structured data. It provides a reliable and query able source for our BI tool.

  The table (SentimentData) was created with a schema designed to match the ASA output, ensuring data integrity.

3.4. Phase 4: Visualization

- **Component: Google Looker Studio**

  A live dashboard was built using the free, web-based BI tool, Google Looker Studio.

It connects directly to the Azure SQL Database using the Microsoft SQL Server connector, allowing it to query the SentimentData table.

The dashboard is configured to auto-refresh, providing a near real-time view of the incoming data.

4. Implementation & Key Challenges

The project's implementation involved local development, cloud resource provisioning, and systematic debugging. Several key challenges were encountered and resolved, demonstrating practical data engineering skills.

- **Initial BI Tool Selection:** The project initially planned to use Power BI. However, a licensing issue with the student account ("User is not licensed for Power BI") prevented the creation of a workspace.
    - **Resolution:** A strategic pivot was made to a more robust and flexible architecture. The output was redirected to an Azure SQL Database, and Google Looker Studio was chosen as the visualization tool. This change ultimately resulted in a more realistic and impressive project stack.
- **Data Producer Connectivity:** The Python script initially failed to connect to the Event Hub, producing a ValueError: Connection string is either blank or malformed.
    - **Resolution:** Debugging revealed that the Windows Command Prompt was misinterpreting special characters in the connection string when passed as an environment variable. The script was re-engineered to read the connection string from a secure config.ini file, a standard industry practice that immediately resolved the issue.
- **Stream Analytics Output Failure:** After configuring the SQL Database output, monitoring charts showed 0 output events. The Activity Log revealed a Failed status with the error "Cannot find table".
    - **Resolution:** Although ASA is designed to create the table automatically, it sometimes fails due to permissions or initialization timing. The issue was resolved by manually creating the SentimentData table in the Azure SQL Database using the Query Editor. This provided a ready-made target for ASA, and upon restarting the job, data began to flow successfully.

5. Results & Showcase

The project successfully achieved all its objectives, culminating in a fully functional, real-time sentiment analysis dashboard.

The final dashboard provides several key insights at a glance:

- **Overall Market Mood:** A pie chart shows the real-time distribution of Positive, Negative, and Neutral sentiment, giving a high-level view of the market.
- **Company-Specific Analysis:** A bar chart compares companies based on the volume of news (Record Count) and the average sentiment (compound_score), allowing for quick identification of market leaders and laggards.
- **Headline Volume Over Time:** A time-series chart visualizes the number of news headlines being processed, helping to identify periods of high market activity.
- **Live Headline Ticker (Not Pictured):** A table can be included to show the latest headlines as they arrive, sorted in descending order of time.

5.1. Actionable Insights

The dashboard is designed not just for monitoring, but for enabling data-driven actions for various stakeholders:

- **For a Trader or Investor:**

  Identify Momentum Shifts: A sudden spike in negative sentiment for a specific company (e.g., "QuantumBank") could be an early warning signal to review a long position or investigate shorting opportunities.

  Spot Emerging Opportunities: A consistent rise in positive sentiment for a lesser-known company (e.g., "FutureVest") could indicate a breakout potential, prompting further due diligence.

  Validate Market Trends: Use the overall market mood pie chart to confirm if a broad market downturn or upturn is being driven by widespread negative or positive news flow.

- **For a Corporate Strategy or PR Team:**

  Competitor Monitoring: The "Company-Specific Analysis" chart allows for real-time benchmarking of the company's public perception against its key competitors. A dip in their sentiment is a potential competitive advantage.

  Crisis Management: The live headline ticker acts as an early warning system. Seeing a negative headline appear for their own company allows the PR team to get ahead of the story and formulate a response immediately, rather than hours later.

- **For a Product Manager:**

  Gauge Product Launch Reception: By filtering for headlines related to a new product launch, a product manager can get an immediate, unfiltered view of its reception in the market.

Identify Market Needs: Consistent neutral or negative news about a specific sector problem (e.g., "service outages") could highlight an opportunity to develop a product that solves that pain point.

6. Conclusion & Future Improvements

This project successfully demonstrates the construction of a modern, real-time data pipeline using a combination of Python, Microsoft Azure, and Google Looker Studio. It showcases key data engineering skills, including data ingestion, stream processing, cloud database management, and business intelligence. The ability to overcome practical challenges highlights a robust and adaptable problem-solving approach.

6.1. Key Learnings

- Proficiency in provisioning and configuring core Azure data services.
- Practical experience in debugging and monitoring a live data pipeline.
- Understanding of the architectural trade-offs between different BI and storage solutions.
- Application of NLP techniques for real-time text analysis.

6.2. Potential Enhancements

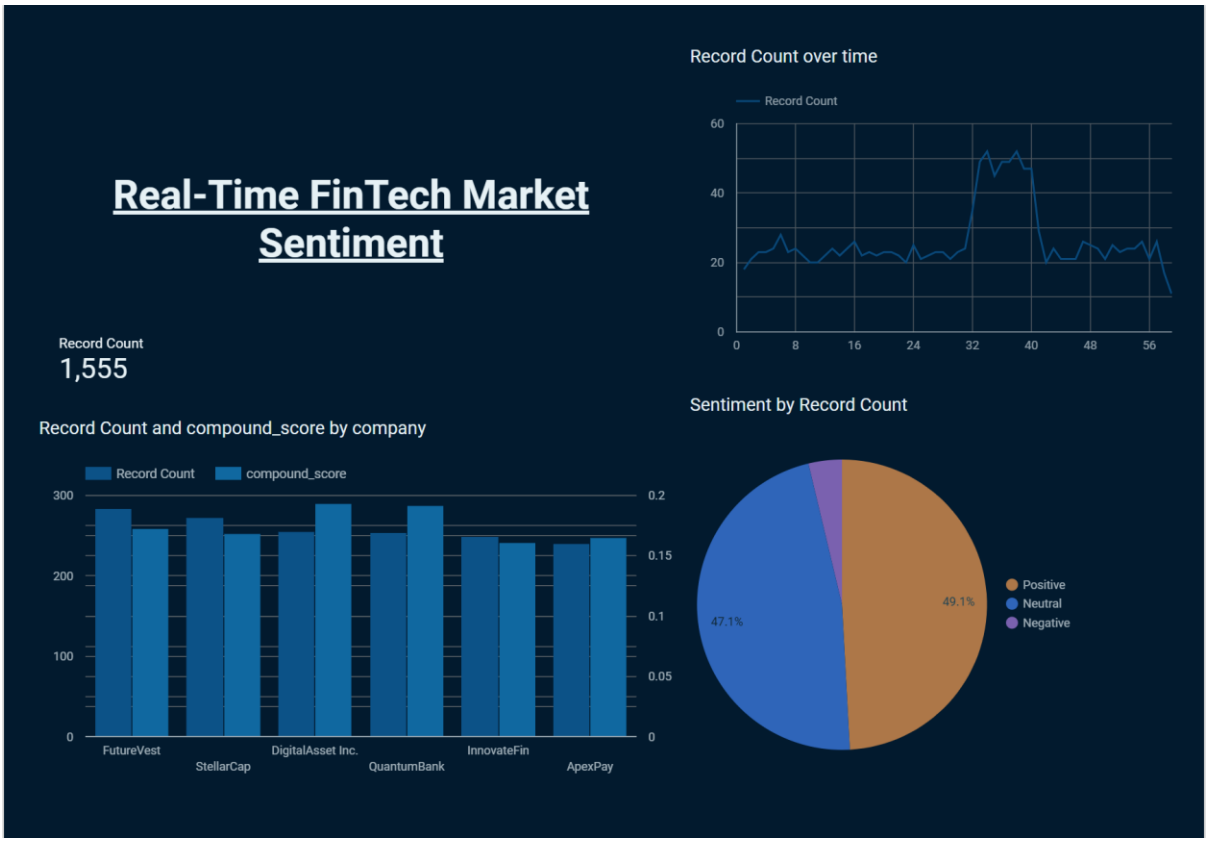
- **Integrate a Real News API:** Replace the synthetic data producer with a live feed from a service like NewsAPI.org or Alpaca for real-world analysis.
- **Deploy the Producer Script:** Move the Python script to a serverless Azure Function or a containerized service to enable continuous, 24/7 data generation without a local machine.
- **Advanced NLP:** Implement more sophisticated NLP models (e.g., fine-tuned transformer models like FinBERT) for more nuanced and accurate sentiment and topic analysis.

7. Appendix

- **Source Code:** The complete source code for the Python producer and project setup files are available on GitHub: [Link to Your GitHub Repository]
- **SQL Table Schema:**
- CREATE TABLE SentimentData (
- EventTimestamp DATETIME2,
- headline NVARCHAR(MAX),
- company NVARCHAR(MAX),

- sentiment NVARCHAR(50),
- compound_score FLOAT
-);

Screenshots of Dashboard



```
Starting data producer... Press Ctrl+C to stop.
Sent: Breaking: StellarCap stock soars as it gets regulatory approval. | Sentiment: Positive
Sent: Alert: The Bank of England probes FutureVest after it under investigation by FCA. | Sentiment: Positive
Sent: Breaking: FutureVest stock soars as it announces record profits. | Sentiment: Positive
Sent: News: FutureVest releases quarterly report this week. | Sentiment: Neutral
Sent: Alert: The Bank of England probes ApexPay after it service outage affects millions. | Sentiment: Positive
Sent: Alert: UK Treasury probes ApexPay after it announces layoffs. | Sentiment: Positive
Sent: News: InnovateFin attends global finance summit this week. | Sentiment: Neutral
Sent: News: FutureVest appoints new CTO this week. | Sentiment: Neutral
Sent: News: InnovateFin releases quarterly report this week. | Sentiment: Neutral
Sent: Breaking: StellarCap stock soars as it partners with tech giant. | Sentiment: Neutral
Sent: Alert: The FCA probes StellarCap after it under investigation by FCA. | Sentiment: Positive
Sent: News: QuantumBank appoints new CTO this week. | Sentiment: Neutral
Sent: Breaking: DigitalAsset Inc. stock soars as it partners with tech giant. | Sentiment: Neutral
Sent: News: StellarCap appoints new CTO this week. | Sentiment: Neutral
Sent: Breaking: ApexPay stock soars as it announces record profits. | Sentiment: Positive
Sent: Breaking: FutureVest stock soars as it gets regulatory approval. | Sentiment: Positive
Sent: News: DigitalAsset Inc. attends global finance summit this week. | Sentiment: Neutral
Sent: Breaking: StellarCap stock soars as it partners with tech giant. | Sentiment: Neutral
Sent: Breaking: QuantumBank stock soars as it gets regulatory approval. | Sentiment: Positive
Sent: Alert: The FCA probes DigitalAsset Inc. after it under investigation by FCA. | Sentiment: Positive
Sent: News: InnovateFin attends global finance summit this week. | Sentiment: Neutral
Sent: Alert: UK Treasury probes StellarCap after it reports unexpected losses. | Sentiment: Positive
Sent: News: FutureVest attends global finance summit this week. | Sentiment: Neutral
Sent: News: StellarCap attends global finance summit this week. | Sentiment: Neutral
Sent: Alert: The Bank of England probes ApexPay after it under investigation by FCA. | Sentiment: Positive
Sent: News: StellarCap releases quarterly report this week. | Sentiment: Neutral
Sent: News: FutureVest releases quarterly report this week. | Sentiment: Neutral
Sent: News: FutureVest attends global finance summit this week. | Sentiment: Neutral
Sent: News: InnovateFin updates terms of service this week. | Sentiment: Neutral
Sent: Alert: The Bank of England probes FutureVest after it announces layoffs. | Sentiment: Positive
Sent: Alert: The Bank of England probes DigitalAsset Inc. after it faces data breach inquiry. | Sentiment: Positive
Sent: News: QuantumBank releases quarterly report this week. | Sentiment: Neutral
Sent: News: InnovateFin appoints new CTO this week. | Sentiment: Neutral
Sent: Breaking: FutureVest stock soars as it announces record profits. | Sentiment: Positive
Sent: News: DigitalAsset Inc. updates terms of service this week. | Sentiment: Neutral
```

Azure Activities

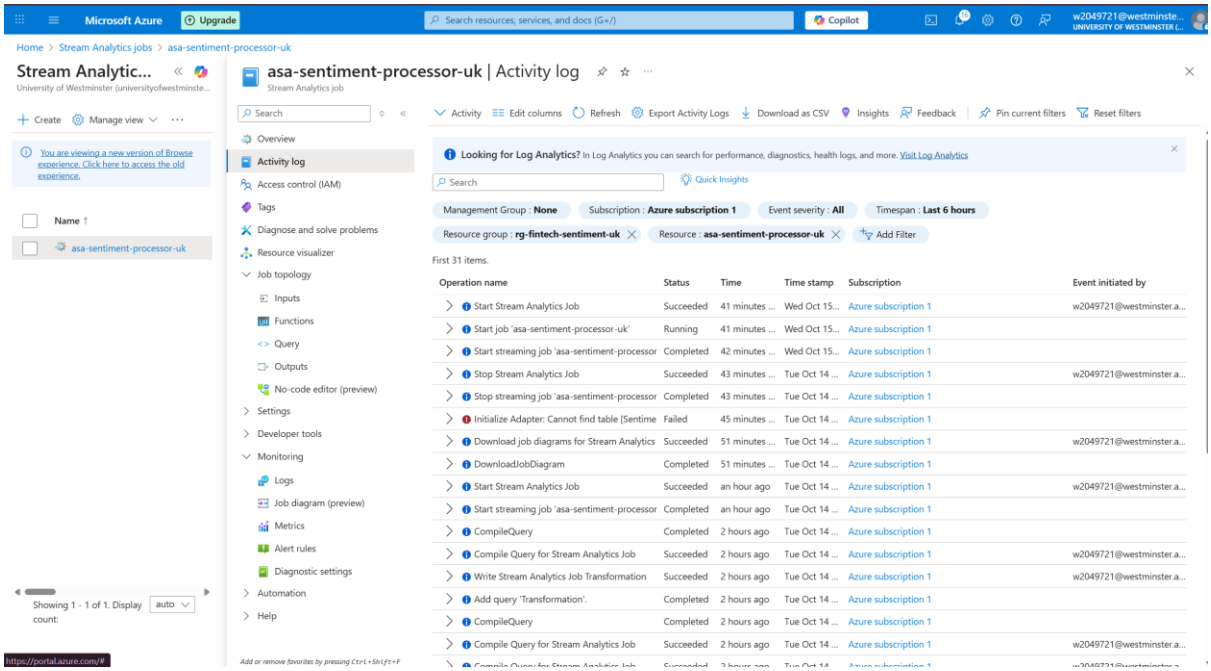


Figure 1 : Activity Log

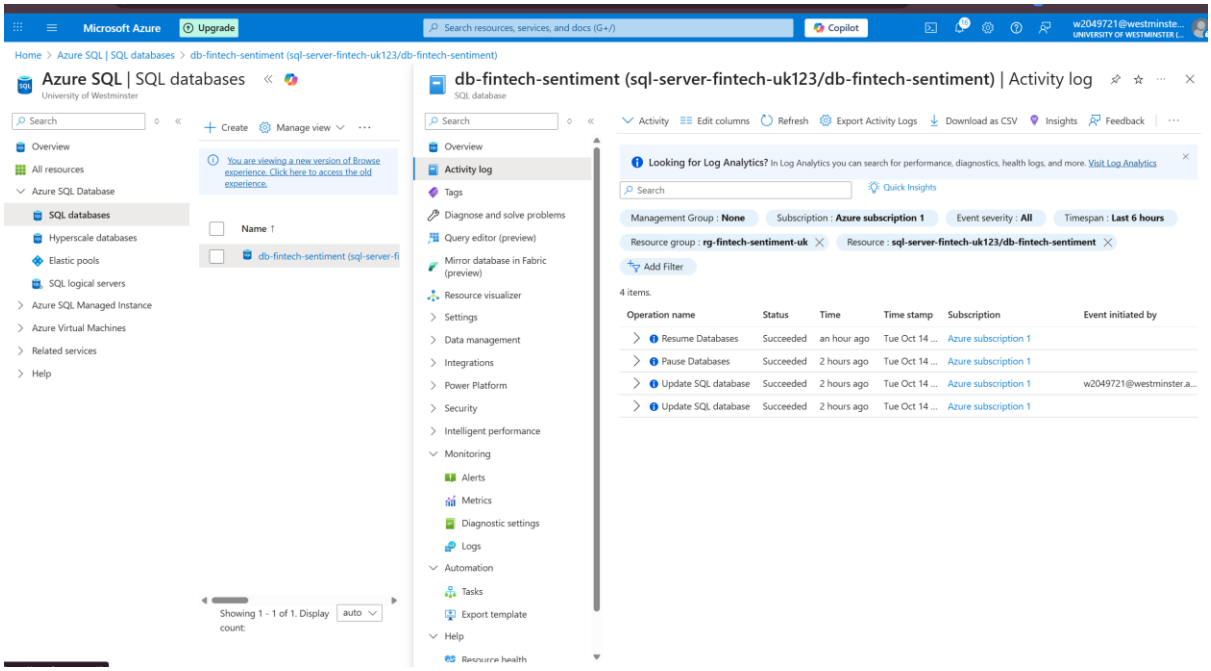


Figure 2: Azure Database log

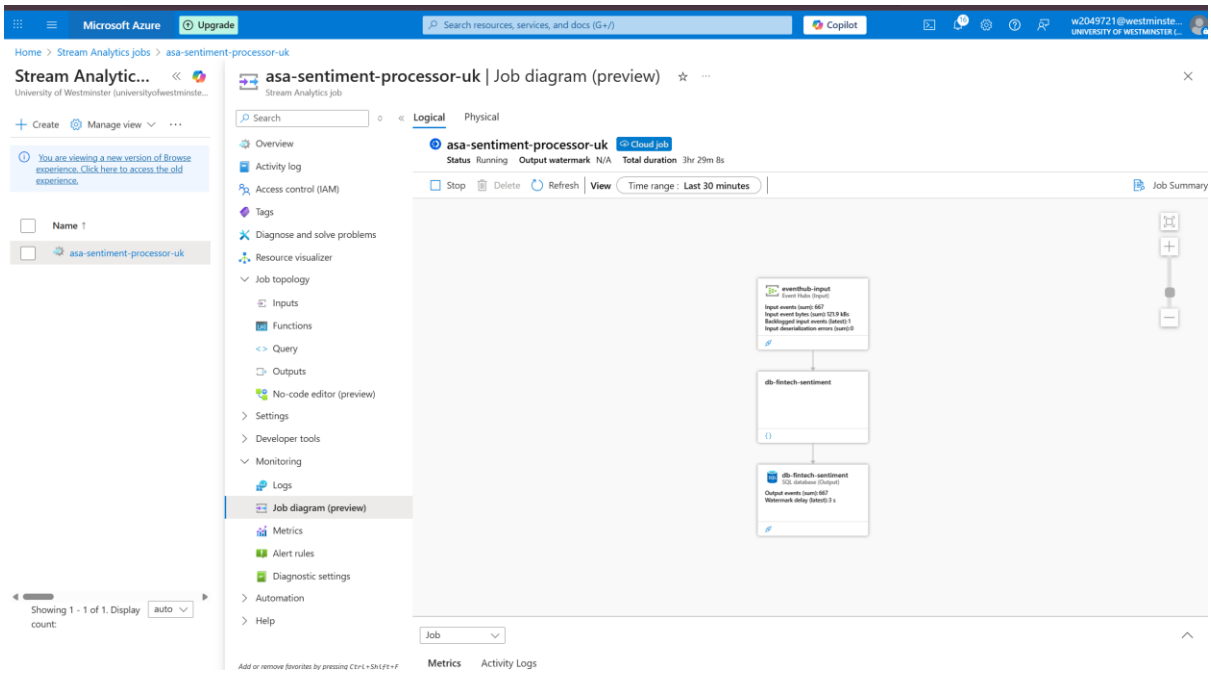


Figure 3 : Streaming Job Workflow

Producer.py

```
import os
import time
import json
```

```
import random
import asyncio
import configparser  # <-- Reads the config file
from faker import Faker
from azure.eventhub import EventData
from azure.eventhub.aio import EventHubProducerClient
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# --- Configuration ---
# Reads the connection string from the config.ini file
config = configparser.ConfigParser()
config.read('config.ini')
EVENT_HUB_CONNECTION_STR = config['azure_event_hub']['connection_string']
EVENT_HUB_NAME = "eh-news-headlines"

# Initialize Faker for synthetic data generation
fake = Faker()

# Initialize NLTK's VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# --- Data Generation Logic (No changes needed here) ---
COMPANIES = ["InnovateFin", "QuantumBank", "ApexPay", "StellarCap",
"FutureVest", "DigitalAsset Inc."]
POSITIVE_EVENTS = ["announces record profits", "secures major funding",
"launches new AI platform", "partners with tech giant", "gets regulatory approval"]
NEGATIVE_EVENTS = ["faces data breach inquiry", "reports unexpected losses",
"under investigation by FCA", "announces layoffs", "service outage affects millions"]
NEUTRAL_TOPICS = ["releases quarterly report", "updates terms of service",
"attends global finance summit", "appoints new CTO"]
```

```
REGULATORS = ["The FCA", "The Bank of England", "UK Treasury"]

def generate_headline():
    """Generates a synthetic Fin-Tech news headline and analyzes its sentiment."""
    headline_type = random.choice(['positive', 'negative', 'neutral'])

    if headline_type == 'positive':
        company = random.choice(COMPANIES)
        event = random.choice(POSITIVE_EVENTS)
        headline = f"Breaking: {company} stock soars as it {event}."
    elif headline_type == 'negative':
        company = random.choice(COMPANIES)
        event = random.choice(NEGATIVE_EVENTS)
        regulator = random.choice(REGULATORS)
        headline = f"Alert: {regulator} probes {company} after it {event}."
    else: # neutral
        company = random.choice(COMPANIES)
        topic = random.choice(NEUTRAL_TOPICS)
        headline = f"News: {company} {topic} this week."

    sentiment_scores = analyzer.polarity_scores(headline)
    compound_score = sentiment_scores['compound']

    if compound_score >= 0.05:
        sentiment = "Positive"
    elif compound_score <= -0.05:
        sentiment = "Negative"
    else:
        sentiment = "Neutral"
```

```
    event_data = {
        'headline': headline,
        'company': company,
        'sentiment': sentiment,
        'compound_score': compound_score,
        'timestamp': time.time()
    }

    return event_data

# --- Main Producer Logic (No changes needed here) ---
async def run():
    """Creates an EventHubProducerClient, sends events in a loop."""
    if not EVENT_HUB_CONNECTION_STR:
        raise ValueError("Connection string is not found in config.ini file. Make sure the file exists and is correct.")

    producer = EventHubProducerClient.from_connection_string(
        conn_str=EVENT_HUB_CONNECTION_STR,
        eventhub_name=EVENT_HUB_NAME
    )

    print("Starting data producer... Press Ctrl+C to stop.")

    try:
        async with producer:
            while True:
                data = generate_headline()
```

```
                event_data_batch = await producer.create_batch()
                event_data_batch.add(EventData(json.dumps(data)))
                await producer.send_batch(event_data_batch)
                print(f"Sent: {data['headline']} | Sentiment: {data['sentiment']}")
                await asyncio.sleep(random.uniform(1, 4))
    except KeyboardInterrupt:
        print("\nStopping data producer.")
    finally:
        print("Producer stopped.")

if __name__ == "__main__":
    asyncio.run(run())
```