Real-Time Market Sentiment and Volatility Tracking: A scalable Pipeline for Cryptocurrency Trades using Kafka, PySpark, and Amazon Redshift

Prepared by:

Michael Hein

Student ID: 300375535

Contents

1.Introduction

Domain Overview: The cryptocurrency market is characterized by extreme volatility and operates 24/7, generating massive volumes of trade data every second. To stay competitive, financial institutions and individual traders require more than just historical data; they need real-time insights to react to market shifts instantly. This research domain focuses on **Event-Driven Architectures (EDA)**, where data is treated as a continuous stream rather than static batches.

Background and Context: Traditional ETL (Extract, Transform, Load) processes often rely on batch processing, which introduces significant latency – often minutes or hours. In the context of digital assets like Bitcoin (BTC) or Ethereum (ETH), such delays can lead to inaccurate risk assessments and missed trading opportunities. The project utilizes the **Coinbase WebSocket API** as a high-velocity data source, feeding into a distributed pipeline built with **Apache Kafka** for ingestion, **PySpark** for stream processing, and **Amazon Redshift** for analytical storage.

2.Problem Statement

The Core Challenge: The primary problem addressed by the research is the **"Insight Latency"** found in traditional data warehousing environments. Specifically, how can we build a system that can ingest, process and analyze thousands of concurrent trade events per second without data loss or significant delay?

Specific Questions to Address:

- How can **Kafka** serve as a fault-tolerant buffer to handle sudden spikes in crypto market activity during periods of high volatility?
- To what extent can **PySpark Structured Streaming** perform real-time windowed aggregations (like moving averages) on "out-of-order" data common in WebSocket feeds?
- How can **Amazon Redshift** be optimized to store and query this high-velocity time-series data for downstream monitoring and alerting?

Solving these questions is vital for developing robust **monitoring and alerting systems** that can detect "flash crashes" or anomalous trading patterns in real-time. This project provides a practical implementation of a scalable, production-grade architecture that bridges the gap between raw data generation and actionable intelligence.

3. Proposed Research Project

3.1 Research Design and Methodology Justification

The research follows an **Event-Driven Architecture (EDA)** design, optimized for high-velocity financial data. The methodology utilizes a Kafka Architecture patten, which simplifies real-time processing by treating all data as a continuous stream rather than maintaining separate batch and speed layers.

Justification:

- **Apache Kafka:** Acts a fault-tolerant distributed buffer, decoupling the high-frequency Coinbase feed from downstream processing. The ensures that no data is lost during market "spikes" or system maintenance.
- **PySpark Structured Streaming:** Chosen for its ability to handle micro-batching and windowed aggregations. Unlike standard Python scripts, Spark can scale horizontally to process thousands of trades per second across a cluster.
- **Amazon Redshift:** Selected for its **Massively Parallel Processing (MPP)** and column storage, which allow for complex SQL queries on millions of historical trades with sub-second response times.

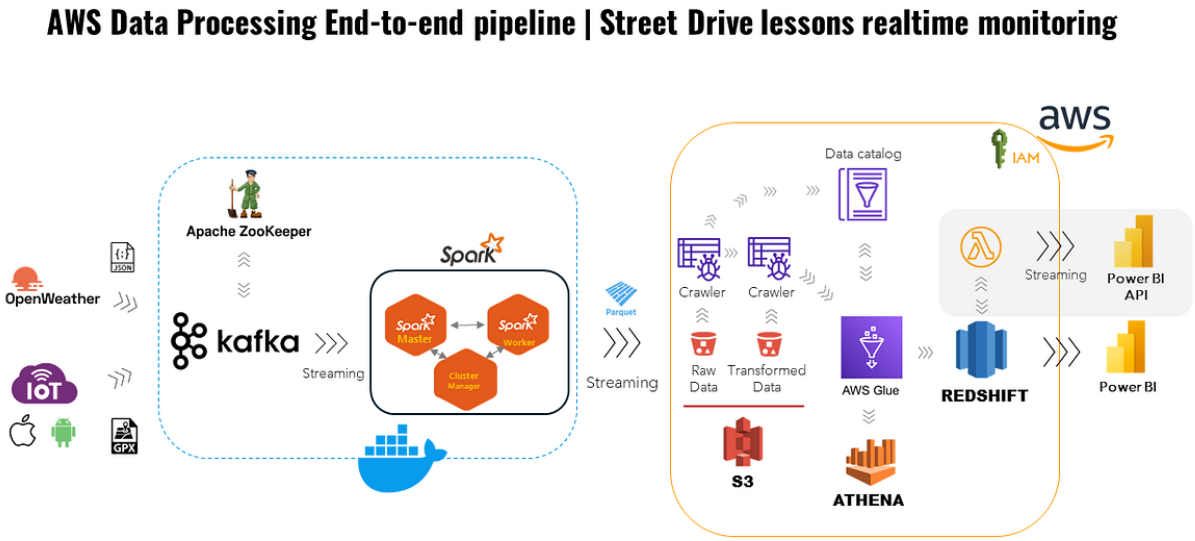
3.2 Data Collection and Analysis Techniques

- **Data Source:** Real-time trade data will be ingested via the **Coinbase WebSocket API** using the ticker and matches channels.
- **Sample Size:** The project will target high-volume trading pairs (e.g., BTC-USD, ETH-USD) over a continuous **7-day period** to capture various market cycles.
- **Analysis Techniques:**
    - **Windowed Aggregations:** Using 1-minute and 5-minute **sliding windows** to calculate Moving Averages and Volume Weighted Average Price (VWAP).
    - **Volatility Detection:** Calculating standard deviation of price changes in real-time to identify "flash crash" patterns.
    - **Watermarking:** Implementing a 10-minute watermark to handle late-arriving data and ensure stateful aggregation accuracy.

3.3 Technology Stack

- **Operating System/Platform: macOS** (Development) and **AWS (Amazon Web Services)** for production deployment (EC2, MSK, Redshift).
- **Programming Languages/Frameworks: Python 3.11, PySpark 3.5,** and **Docker Compose** for environment orchestration.
- **Database: Amazon Redshift** (Data Warehouse) and S3 (Staging).
- **Backend: Apache Kafka** (Message Broker) and **Spark Streaming** (Processing Engine).

- **Frontend: Grafana** or **Amazon Quick Sight** for real-time dashboarding of trade metrics.



3.4 Expected Results and Practical Applications

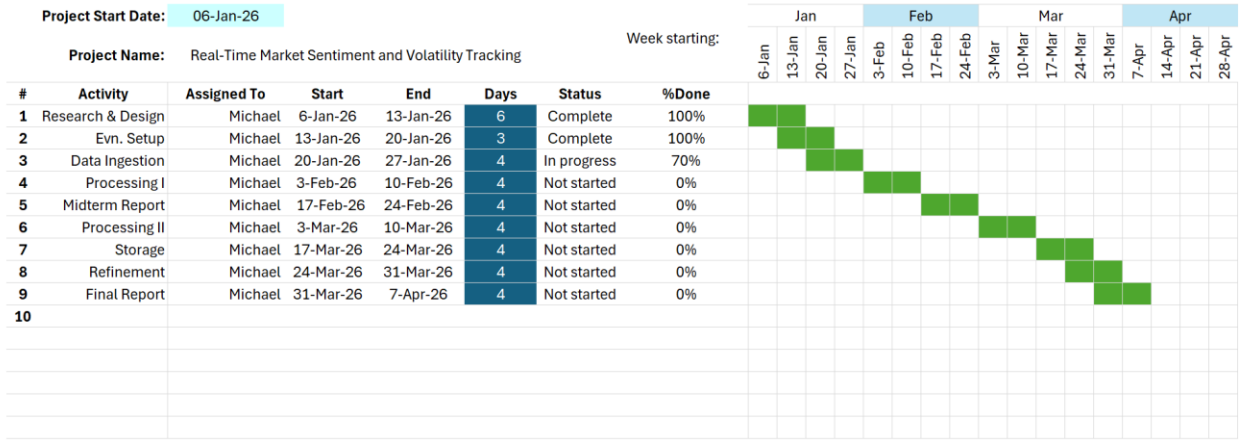
- **Reduce Latency:** The system is expected to achieve sub-second latency from the moment a trade occurs on Coinbase to its visibility in the analytical dashboard.
- **Accurate Risk Monitoring:** The pipeline will demonstrate the ability to trigger **automated alerts** when volatility exceeds pre-defined thresholds.
- **Practical Contribution:** This research provides a blueprint for financial institutions to move away from slow "End-of-Day" reporting toward **Active Risk Management,** allowing for immediate responses to market manipulation or liquidity crises.

4. Project Planning and Timelines

Week	Phase	Milestone & Deliverables	Estimated Hours
02	**Research & Design**	Finalize Proposal; Research Coinbase WebSocket & Redshift schemas.	12 hrs
3	**Env. Setup**	Configure Docker, Kafka, and Spark Master/Worker containers.	10 hrs
4	**Data Ingestion**	Build producer.py to stream live BTC/ETH trades into Kafka.	15 hrs
5	**Processing I**	Develop PySpark script for basic stream consumption and cleaning.	15 hrs
6	**Midterm**	**Deliverable: Midterm Report & 10-min Video Demo.**	12 hrs
7	**Processing II**	Implement windowed aggregations (VWAP, Moving Averages).	15 hrs

8	**Storage**	Configure Redshift connector and implement data sinking logic.	12 hrs
9	**Refinement**	Implement real-time alerting and perform performance testing.	15 hrs
10	**Finalization**	**Deliverable: Final Report & Oral Defense Presentation.**	14 hrs

Gantt Chart



Project Start Date: 06-Jan-26

Project Name: Real-Time Market Sentiment and Volatility Tracking

Week starting: Jan | Feb | Mar | Apr — 6-Jan, 13-Jan, 20-Jan, 27-Jan, 3-Feb, 10-Feb, 17-Feb, 24-Feb, 3-Mar, 10-Mar, 17-Mar, 24-Mar, 31-Mar, 7-Apr, 14-Apr, 21-Apr, 28-Apr

#	Activity	Assigned To	Start	End	Days	Status	%Done
1	Research & Design	Michael	6-Jan-26	13-Jan-26	6	Complete	100%
2	Evn. Setup	Michael	13-Jan-26	20-Jan-26	3	Complete	100%
3	Data Ingestion	Michael	20-Jan-26	27-Jan-26	4	In progress	70%
4	Processing I	Michael	3-Feb-26	10-Feb-26	4	Not started	0%
5	Midterm Report	Michael	17-Feb-26	24-Feb-26	4	Not started	0%
6	Processing II	Michael	3-Mar-26	10-Mar-26	4	Not started	0%
7	Storage	Michael	17-Mar-26	24-Mar-26	4	Not started	0%
8	Refinement	Michael	24-Mar-26	31-Mar-26	4	Not started	0%
9	Final Report	Michael	31-Mar-26	7-Apr-26	4	Not started	0%
10							

5.AI Use Section

AI Tool Name	**Version, Account Type**	**Specific feature for which the AI tool was used**	**Value Addition**
Gemini	Advanced (Pro)	Architectural Design: Brainstorming the integration of Kafka, PySpark, and Redshift for crypto data.	I defined the specific business use case (crypto volatility) and selected the Coinbase WebSocket as the primary data source.
Gemini	Advanced (Pro)	Technical Setup: Generating Docker file, docker-compose.yml, and requirements.txt for Mac.	I performed the manual installation and troubleshooting on my local Mac terminal to verify the environment was functional.
Google Antigravity	Advanced (Pro)	IDE for python code	I wrote AI assisted codes in the IDE

6. Work Log Table

Date	Number of Hours	Description of work done
2026-01-16	3	Configured project environment by creating Docker file, docker-compose.yml, devcontainer. Json vis Mac terminal to ensure full-service isolation for Kafka and Spark
2026-01-16	4	Researched and initialized requirement.txt with essential dependencies with pyspark, Kafka and amazon redshift integration
2026-01-16	4	Researched and wrote the introduction, problem statement, proposed research project sections of the Project Proposal
2026-01-20	4	Designed project planning and timelines. Watched YouTube video on how to create Gantt chart on excel. Created the Gantt chart.
2026-01-20	1.5	Watched YouTube videos about Kafka

7. References

1. Fundamentals of Data Engineering Masterclass https://www.youtube.com/watch?v=hf2go3E2m8g
2. Learn Kafka in 10 minutes https://www.youtube.com/watch?v=Ydts27Qa8H0
3. Stock Market Real-time Data Analysis Using Kafka https://www.youtube.com/watch?v=KerNf0NANMo
4. Building a Real-Time Data Pipeline with PySpark, Kafka, and Redshift https://www.youtube.com/watch?v=iluXuIM-als

5. Real-Time Data Pipeline with Apache Kafka and AWS
   https://www.youtube.com/watch?v=G87fm-tjhmY
6. Fundamentals of Data Engineering: Plan and Build Robust Data Systems book by Joe Reis
   https://www.amazon.ca/Fundamentals-Data-Engineering-Robust-Systems/dp/1098108302

8. Appendix: Prompt History

- How do I create real-time data pipelines?
- What is the usefulness of Kafka, spark and amazon redshift in real-time data streaming?
- What are the best open-source APIs for real time data streaming?