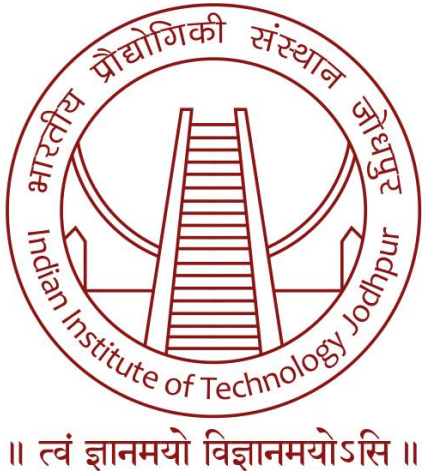
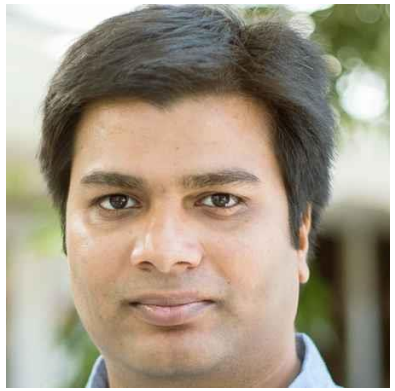
Real-Time Stock Market Data Analysis and Trading Signal Generation using GenAI

TRIMESTER 3: ADVANCED DATA ENGINEERING IN CLOUD



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Submitted To:
Dr. Pradip Sasmal
IIT Jodhpur
Date: 22 July 2024

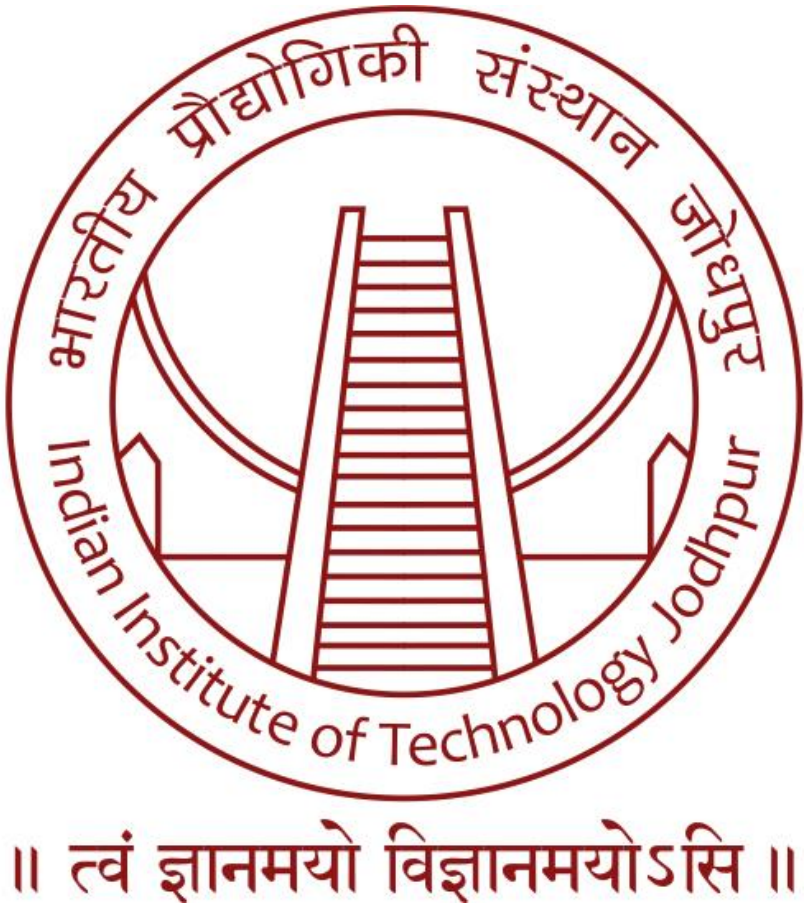


Executed by:

Pooja Yadav
G23AI1026

Preeti Vishvakarma
G23AI1016

Jay Vishvakamra
G23AI1016



Executive Summary

This project aims to perform real-time analysis of stock market data and generate trading signals using Python, Apache Flink, Amazon Kinesis Data Streams, and the Alpaca API for stock trading. The objective is to collect historical and real-time data on stock prices and news, process the data to compute trading signals, and execute trades based on these signals. By integrating sentiment analysis using Python libraries like NLTK and leveraging AWS services for scalable data processing and execution, this project enhances the accuracy and efficiency of trading strategies.

Table of Contents

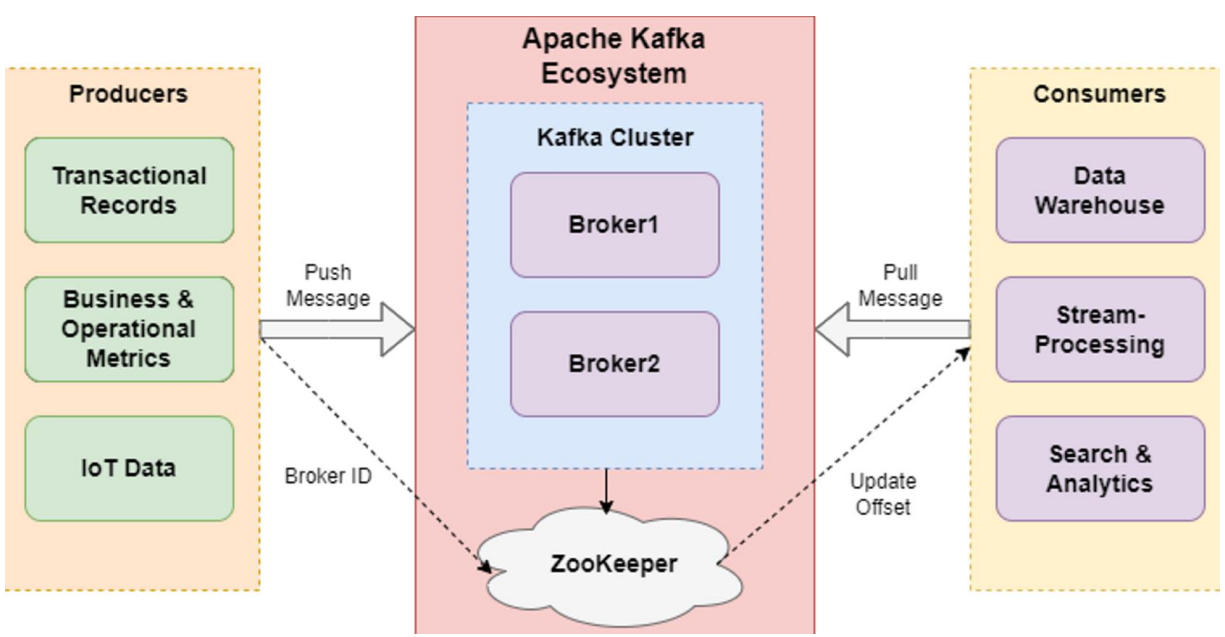
Introduction

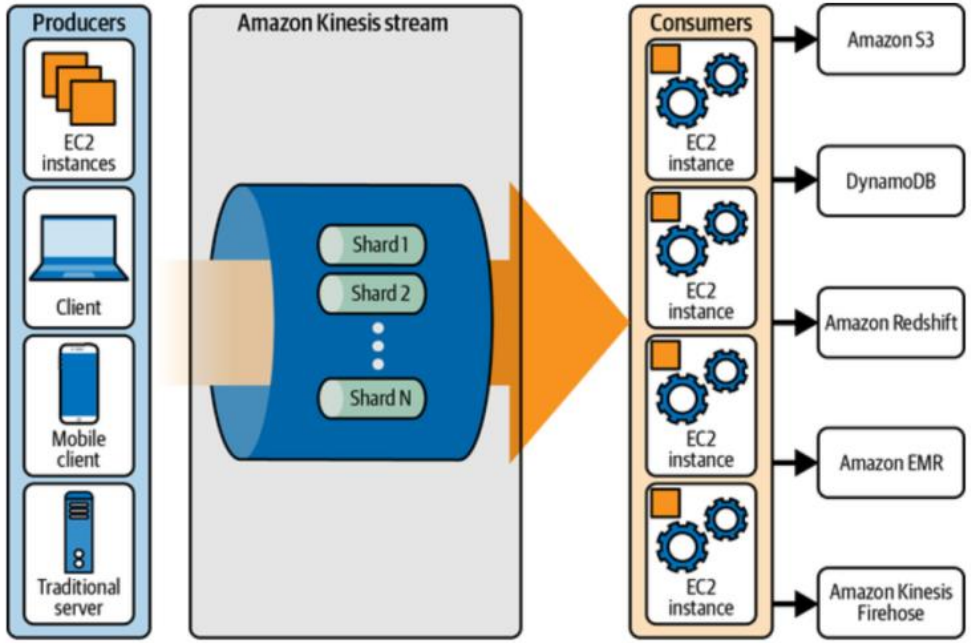
The stock market is characterized by its fast-paced nature and dynamic changes, necessitating real-time data analysis for informed trading decisions. This project utilizes Python along with AWS services such as Amazon Kinesis Data Streams, AWS Lambda, and Amazon ECS to create a scalable and efficient system for real-time stock market data analysis and trading signal generation. By leveraging Python's data processing capabilities and AWS's infrastructure, the project aims to deliver timely and accurate trading signals to enhance profitability.

Literature Review

Existing Solutions

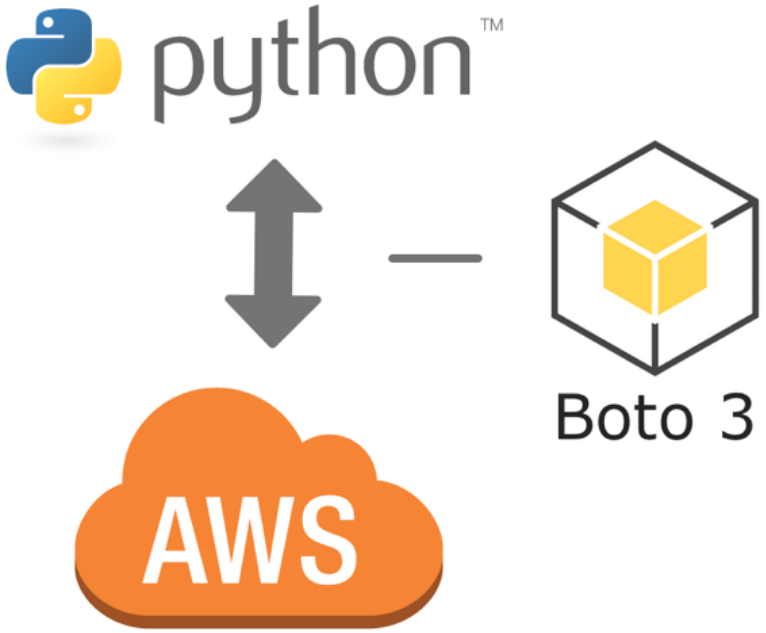
Traditional stock market analysis methods often rely on batch processing, which may not be suitable for real-time trading. AWS provides robust services like Amazon Kinesis Data Streams, designed for real-time data streaming and processing, offering high scalability and low latency.





Python in Financial Applications using GenAI

Python is widely used in financial applications due to its extensive libraries and ease of integration with data sources and APIs. Libraries like Pandas, NLTK for sentiment analysis, and Boto3 for AWS integration make Python a suitable choice for this project.

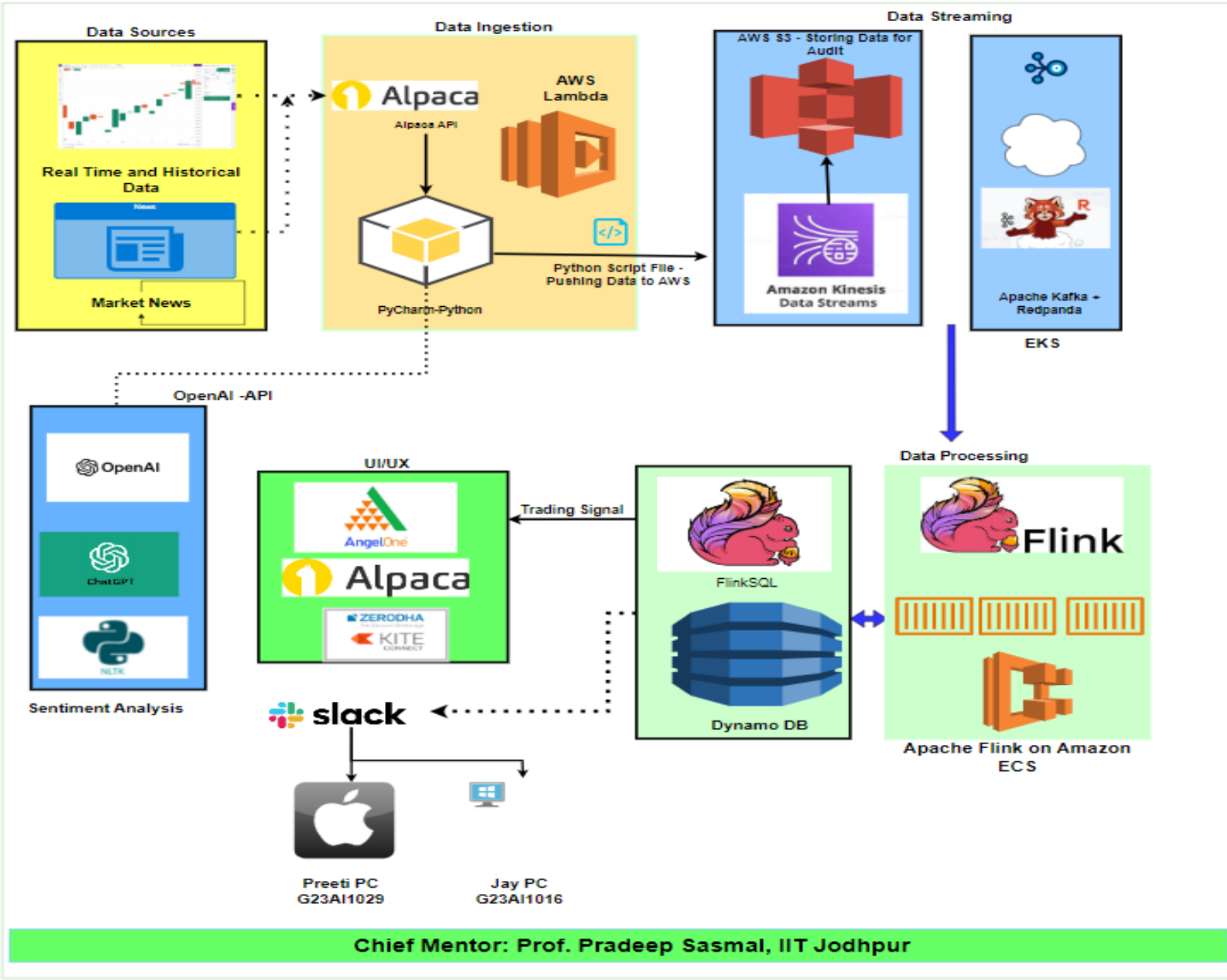


Project Objectives

The primary objectives of this project are:

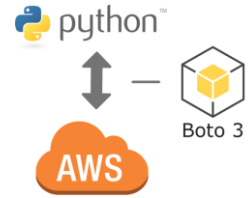
- **Data Collection**: Implement Python scripts to fetch historical and real-time stock prices and news data from APIs (e.g., Alpaca API, Market News APIs).
- **Real-Time Data Processing**: Utilize Apache Flink deployed on Amazon ECS to process streaming data from Amazon Kinesis Data Streams and generate trading signals.
- **Sentiment Analysis**: Integrate NLTK for sentiment analysis on news data within the data processing pipeline to enhance trading signal accuracy.
- **Trade Execution**: Use the Alpaca API to execute buy/sell orders based on the generated trading signals.
- **Scalability**: Deploy scalable solutions using AWS services to handle large volumes of data efficiently.

Hybrid Data Cloud Architecture



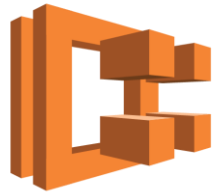
Technology Stack

1. Data Collection



2. Data Streaming

3. Data Processing



4. Sentiment Analysis

Technology Stack

5. Trade Execution

6. User Interface and Notifications

7. Monitoring and Management

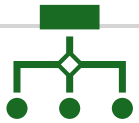
Components

1. Data Sources

Alpaca API:
Fetches historical and real-time stock data.

Market News APIs:

Collects news articles related to stock market movements.



2. Data Ingestion
AWS Lambda:

Python functions fetch data from APIs and push it to Amazon Kinesis Data Streams.
Python Scripts:

Developed in PyCharm, these scripts handle data ingestion, formatting, and pushing to AWS services.

3. Data Streaming

Amazon Kinesis Data Streams:

Collects and manages large streams of data records in real-time.

Redpanda (alternative):

Kafka-compatible streaming platform if needed for compatibility or specific features not covered by Amazon Kinesis.

4. Data Processing
Apache Flink on Amazon ECS:

Real-time data processing engine for computing trading signals based on incoming data streams.

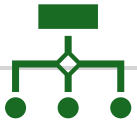
Components

5. Sentiment Analysis

Sentiment Analysis

NLTK: Python library for natural language processing and sentiment analysis on news data, integrated within Apache Flink jobs.

Alternate: ChatGPT



6. Trading Signal Generation

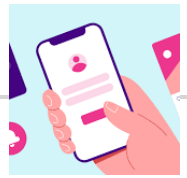
Apache Flink:

Processes sentiment analysis results to generate buy/sell signals based on real-time and historical data.

7. Trade Execution

Alpaca API:

Executes trades based on the generated signals. Python scripts interact with the Alpaca API to place orders.

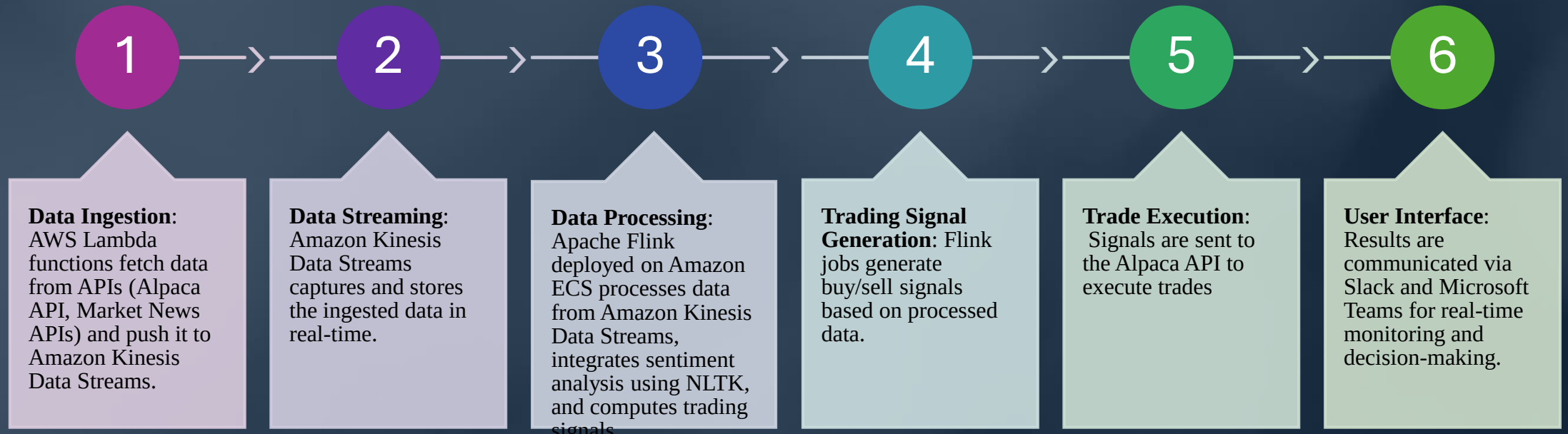


8. Data Processing

Slack/Microsoft Teams:

Notification platforms for displaying trading signals, executed trades, and status updates.

Data Flow


1
2
3
4
5
6
Data Ingestion: AWS Lambda functions fetch data from APIs (Alpaca API, Market News APIs) and push it to Amazon Kinesis Data Streams.
Data Streaming: Amazon Kinesis Data Streams captures and stores the ingested data in real-time.
Data Processing: Apache Flink deployed on Amazon ECS processes data from Amazon Kinesis Data Streams, integrates sentiment analysis using NLTK, and computes trading signals.
Trading Signal Generation: Flink jobs generate buy/sell signals based on processed data.
Trade Execution: Signals are sent to the Alpaca API to execute trades
User Interface: Results are communicated via Slack and Microsoft Teams for real-time monitoring and decision-making.

Implementation

Amazon Kinesis Data Streams: Configure data streams to ingest and manage real-time data from APIs.

AWS Lambda: Implement Python functions for event-driven data ingestion from APIs to Amazon Kinesis Data Streams.

NLTK Integration: Develop Python scripts to perform sentiment analysis on news data within Apache Flink jobs.

Alpaca API Integration: Utilize Python scripts to interact with the Alpaca API for trade execution based on generated signals.

Alpaca API Integration: Utilize Python scripts to interact with the Alpaca API for trade execution based on generated signals.

AWS Lambda Data Ingestion (Python)

```
import boto3
import requests
import json

kinesis_client = boto3.client('kinesis')

def lambda_handler(event, context):
  # Example: Fetch data from Alpaca API
  response = requests.get('https://api.alpaca.markets/v2/stocks/AAPL/quote')
  data = response.json()

  # Push data to Kinesis Data Stream
  response = kinesis_client.put_record(
    StreamName='stock_data_stream',
    Data=json.dumps(data),
    PartitionKey='1'
  )

  print(f"Data pushed to Kinesis Data Stream: {data}")
```

Apache Flink Data Processing with NLTK (Python)

```
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.datastream.connectors import FlinkKinesisProducer
from pyflink.datastream.serialization import SimpleStringSchema
from nltk.sentiment.vader import SentimentIntensityAnalyzer
env = StreamExecutionEnvironment.get_execution_environment()
stream = env.add_source(
FlinkKinesisConsumer(
"stock_data_stream",
SimpleStringSchema(),
properties
)
)
def analyze_sentiment(text):
sia = SentimentIntensityAnalyzer()
sentiment_score = sia.polarity_scores(text)
return sentiment_score['compound']
def process_data(data):
sentiment_score = analyze_sentiment(data['news'])
# Generate trading signals based on sentiment score and stock data
trading_signal = "BUY" if sentiment_score > 0.2 else "SELL"
return f"{data['stock']} - {trading_signal}"
stream.map(process_data).add_sink(
FlinkKinesisProducer(
"result_stream",
SimpleStringSchema(),
properties
)
)
env.execute("StockDataAnalysisJob")
```

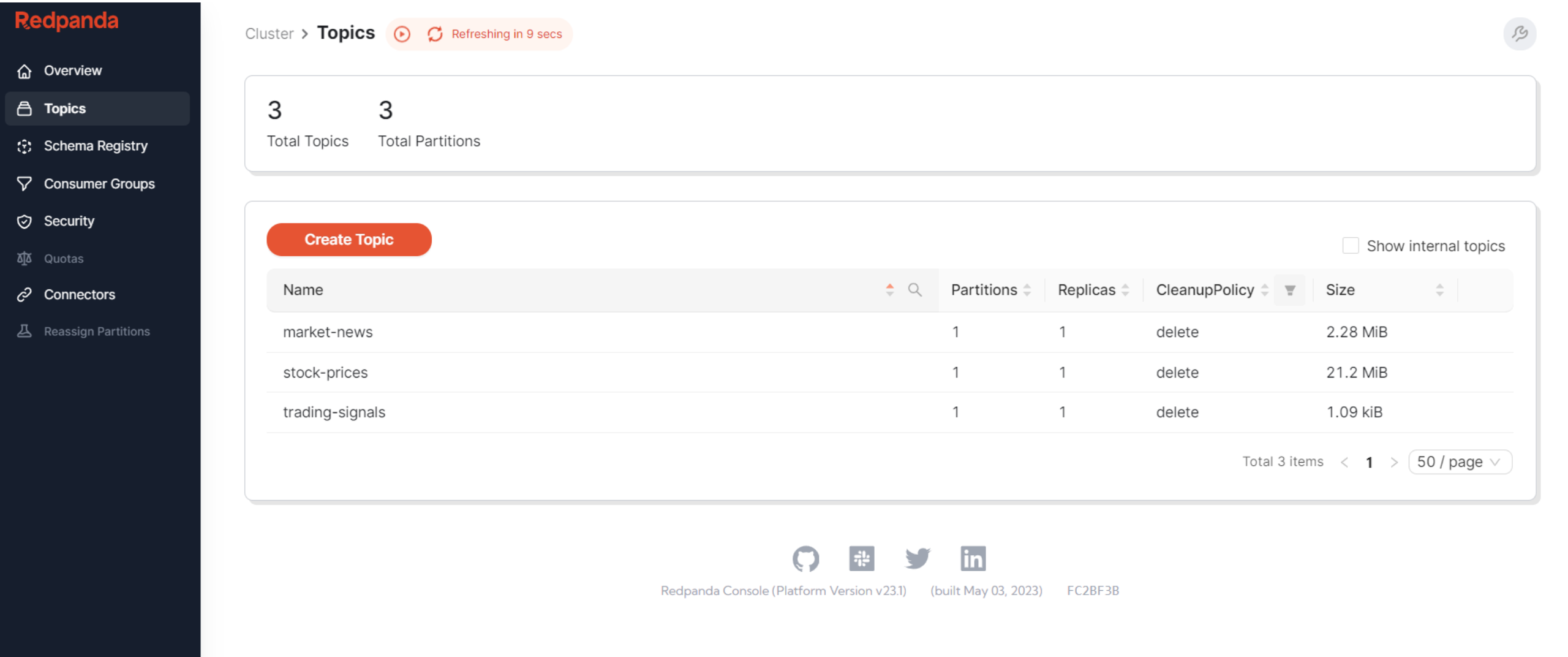
Integration with Alpaca API

```
import alpaca_trade_api as tradeapi
api = tradeapi.REST('APCA-API-KEY-ID', 'APCA-API-SECRET-KEY', base_url='https://paper-api.alpaca.markets')
def execute_trade(symbol, quantity, side):
api.submit_order(
symbol=symbol,
qty=quantity,
side=side,
type='market',
side=side,
time_in_force='gtc' )
# Example usage
execute_trade('AAPL', 10, 'buy')
```

Data Analysis and Results

Data Collection (Market News, Stock Price, Trading Signals)

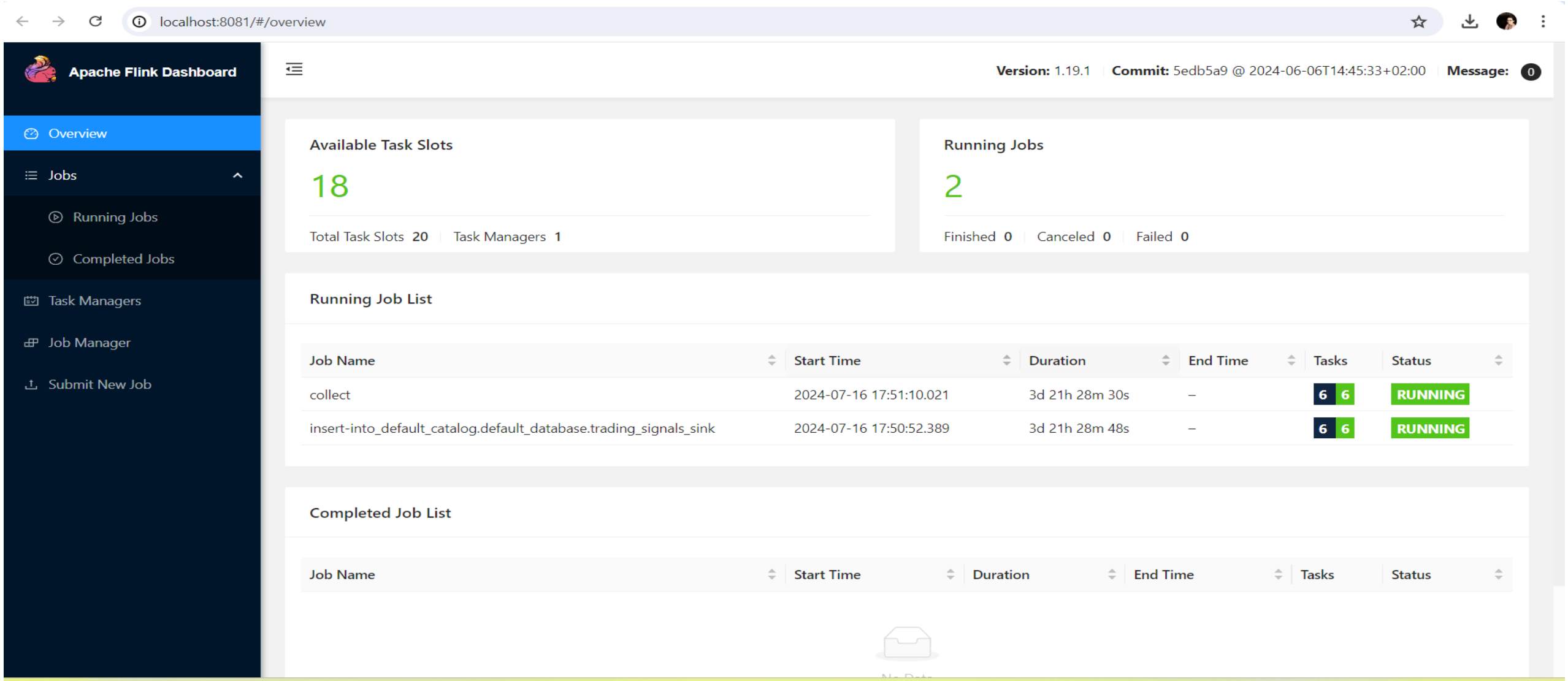
The system collects data from various sources, including stock prices and news articles. Redpanda is used to stream this data in real-time, ensuring that the latest information is always available for analysis.



Data Analysis and Results

Data Processing

Apache Flink processes the data in real-time, leveraging NLTK for sentiment analysis on news data, and generates trading signals based on the processed data.



Apache Flink Dashboard

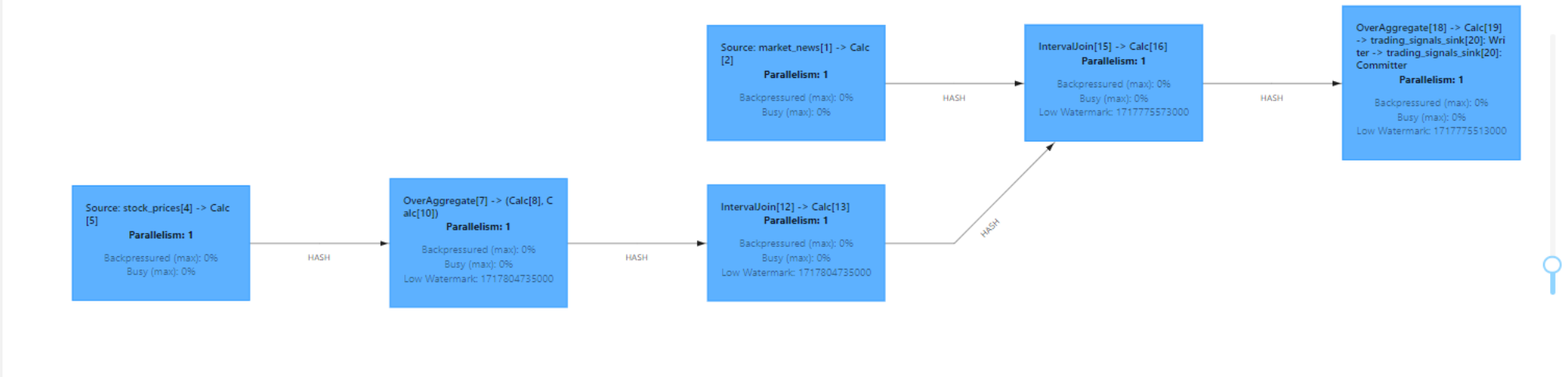
- Overview
- Jobs
  - Running Jobs
  - Completed Jobs
- Task Managers
- Job Manager
- Submit New Job

insert-into_default_catalog.default_database.trading_signals_sink

Cancel Job

Job ID	1f51ba6bd0599154bc0db36b10f7e1af	Job State	RUNNING 6	Actions	Job Manager Log
Start Time	2024-07-16 17:50:52.389	Duration	4d 2h 42m 58s		

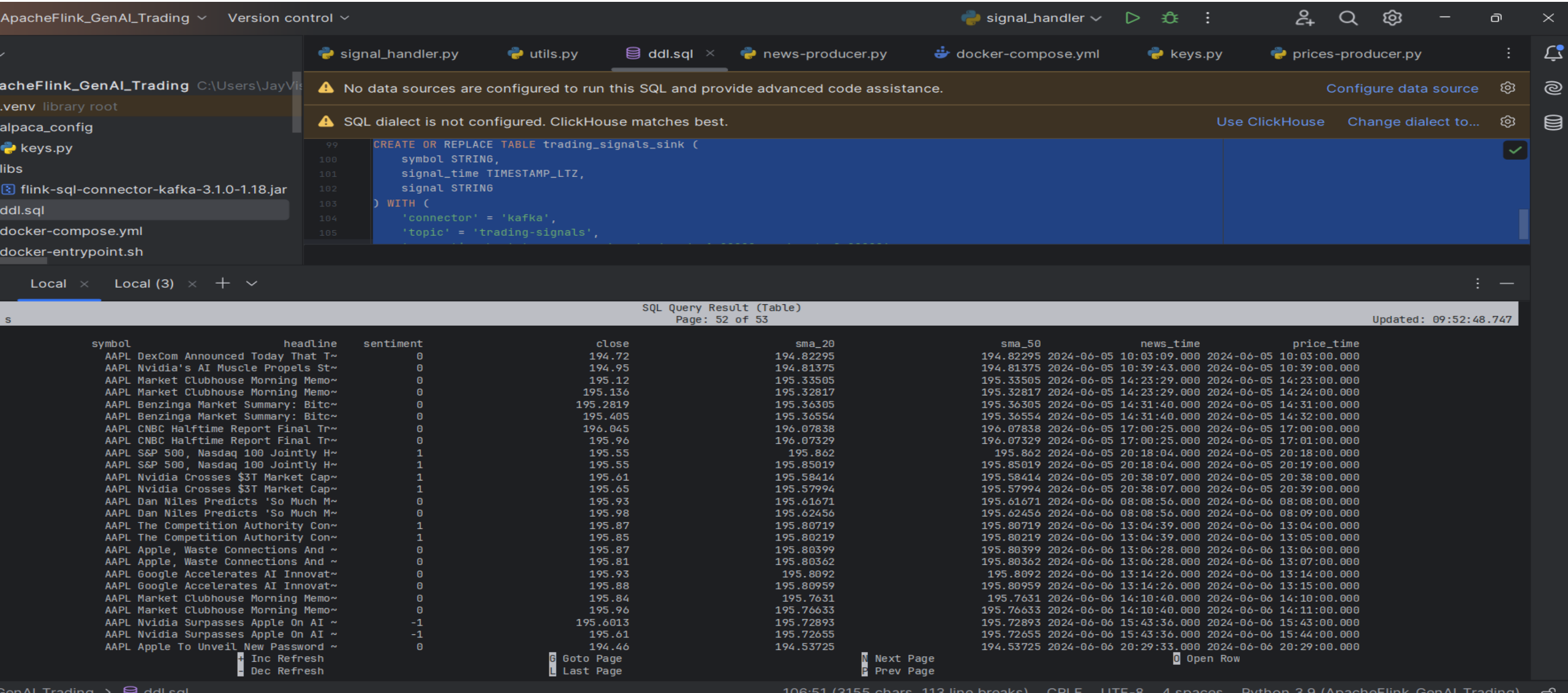
Overview | Exceptions | TimeLine | Checkpoints | Configuration



Data Analysis and Results

Sentiment Analysis

GenAI performs sentiment analysis on news data, providing insights that influence trading decisions. The sentiment scores are integrated into the Flink jobs, enhancing the accuracy of the generated signals.



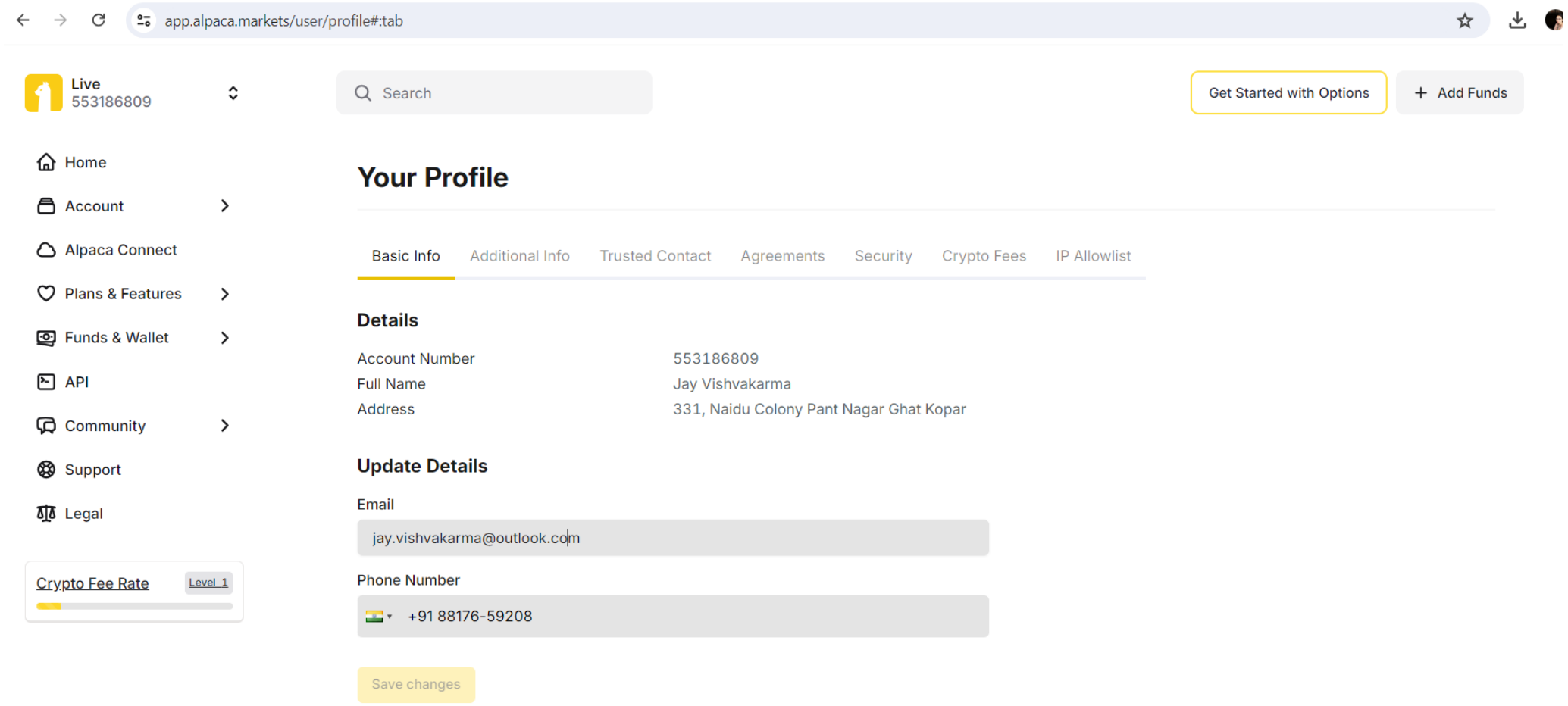
Messages | Consumers | Partitions | Configuration | ACL | Documentation

PARTITION: All | START OFFSET: Newest-50 | MAX RESULTS: 50 | FILTER | Actions | Quick Search

Offset	Partition	Timestamp	Key	Value (Preview)
6	0	7/16/2024, 5:51:09 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-05-31 10:00:00Z","signal":"BUY"}
5	0	7/16/2024, 5:51:08 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-05-08 11:11:07Z","signal":"BUY"}
4	0	7/16/2024, 5:51:08 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-05-06 13:41:10Z","signal":"BUY"}
3	0	7/16/2024, 5:51:05 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-04-12 12:20:49Z","signal":"BUY"}
2	0	7/16/2024, 5:51:04 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-03-27 07:59:10Z","signal":"BUY"}
1	0	7/16/2024, 5:51:04 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-03-22 12:17:52Z","signal":"SELL"}
0	0	7/16/2024, 5:51:02 PM	⊘	+ {"symbol":"AAPL","signal_time":"2024-03-06 20:17:29Z","signal":"BUY"}

Trading Results:

The Alpaca API is used to execute trades based on the generated signals. The performance of the trading strategy is evaluated based on various metrics, including profitability, risk, and execution speed.



Top Positions

Asset Class All | Side All | View All

Asset	Price	Qty	Market Value	Total P/L ($)
AAPL	$224.31	201	$45,086.31	-$2,443.31
ACN	$329.19	199	$65,508.81	+$1,826.82
ADSK	$242.45	100	$24,245	-$1,292
GOOGL	$177.66	100	$17,766	-$975
AAL	$10.58	100	$1,058	-$30

Recent Orders

Asset Class All | Side All | Date Precision | View All

Symbol	Type	Qty	Average Cost	Amount	Status	Date
ADSK	Market BUY	100	$255.37	$25,537	Filled	4 days ago
GOOGL	Market BUY	100	$187.41	$18,741	Filled	4 days ago
AAL	Market BUY	100	$10.88	$1,088	Filled	4 days ago
ACN	Market BUY	199	$320.01	$63,681.99	Filled	4 days ago
AAPL	Market BUY	1	$235.26	$235.26	Filled	4 days ago
AAPL	Market BUY	100	$236.71	$23,671	Filled	7 days ago

Endpoint

https://paper-api.alpaca.markets/v2

Key

PKH98QULS4K8DXYQSDUI

Regenerate

Watchlist

Edit

No assets in watchlist

Challenges and Solutions

Scalability

- AWS services such as Amazon Kinesis Data Streams and Amazon ECS ensure scalability to handle large volumes of data and computation demands efficiently.

Latency

- Optimizing data processing pipelines and leveraging AWS's low-latency services minimize delays in generating trading signals and executing trades.

Cost Management

- Utilizing AWS Lambda for event-driven data ingestion and efficient resource management in ECS helps in cost optimization and scalability.

Conclusion:

This project demonstrates the feasibility of using AWS services and Python for real-time stock market data analysis and trading signal generation.

Future Work:

- Future work involve enhancing trading strategies, integrating additional data sources, and further optimizing system performance using advanced machine learning models.

  Data Enrichment Pipeline
  Integrating ML Model

References:

AWS Documentation - https://aws.amazon.com/documentation/

Apache Flink Documentation - https://flink.apache.org/

Alpaca API Documentation - https://alpaca.markets/docs/api-documentation/

NLTK Documentation - https://www.nltk.org/

OpenAI - https://platform.openai.com/docs/concepts

Thank you