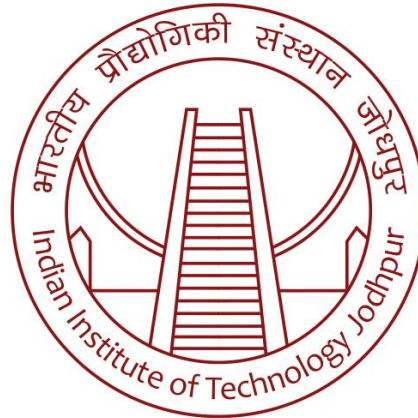


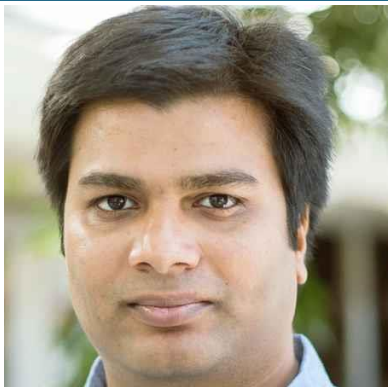
Real-Time Stock Market Data Analysis and Trading Signal Generation using GenAI

TRIMESTER 3: ADVANCED DATA ENGINEERING IN CLOUD



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

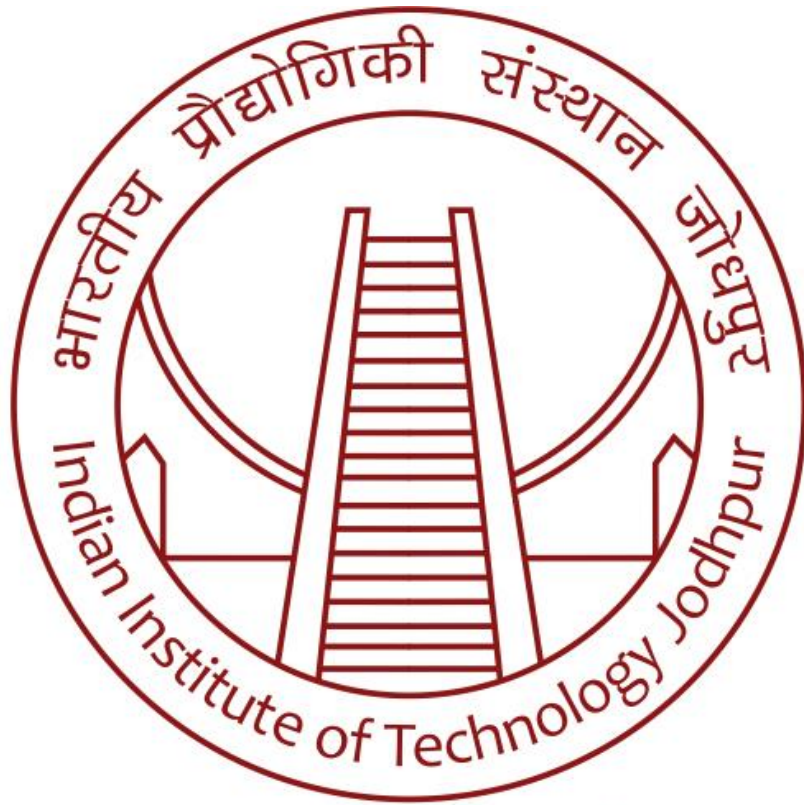
Submitted To:
Dr. Pradip Sasmal
IIT Jodhpur
Date: 22 July 2024



Executed by:

Pooja Yadav	Preeti Vishvakarma	Jay Vishvakamra
G23AI1026	G23AI1016	G23AI1016





॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

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

Literature
Review

Project
Objectives

System
Architecture

Implementation

Data Analysis
and Results

Challenges and
Solutions

Conclusion and
Future Work

References



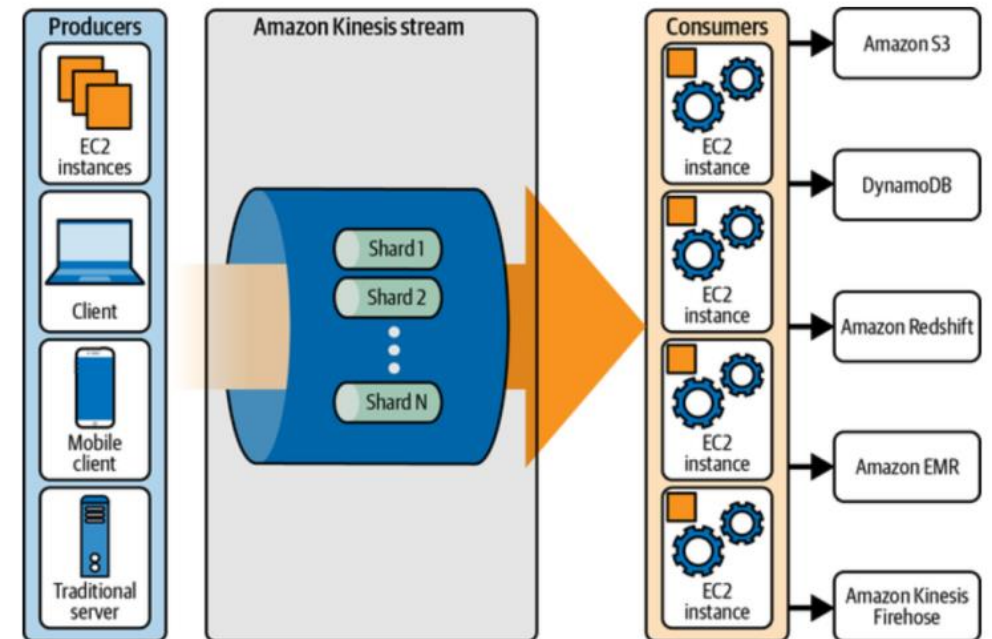
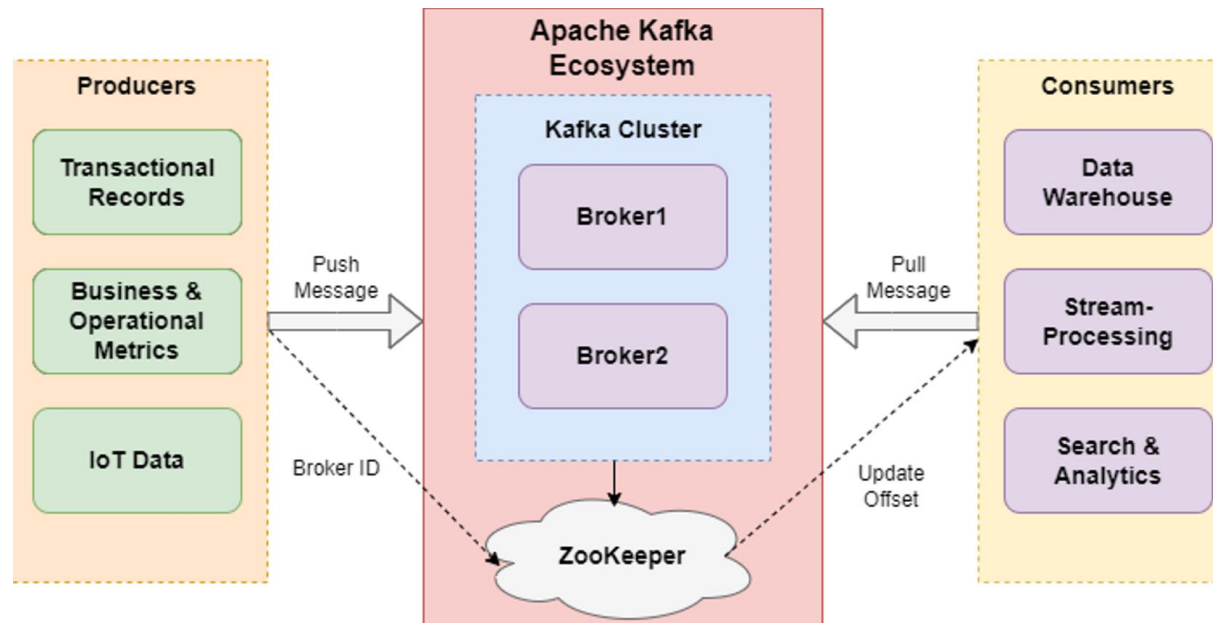
Introduction

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

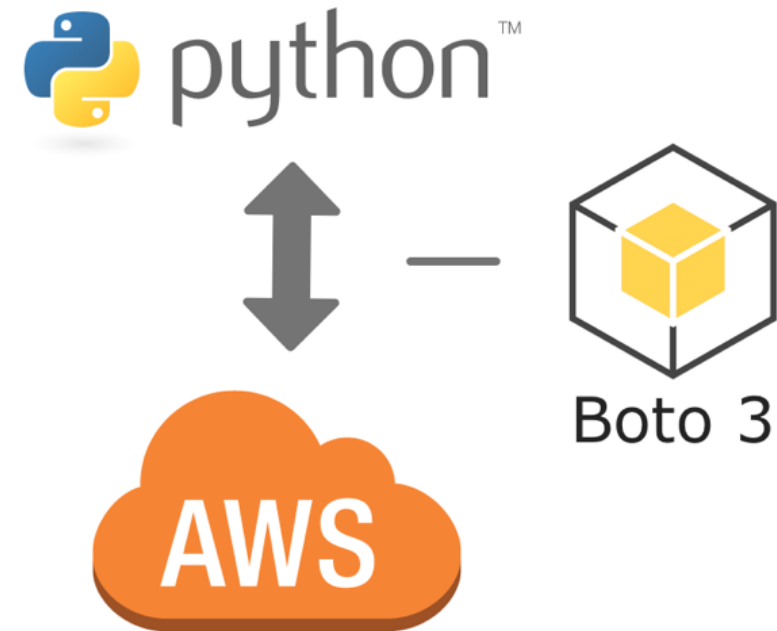
Existing Solutions

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.

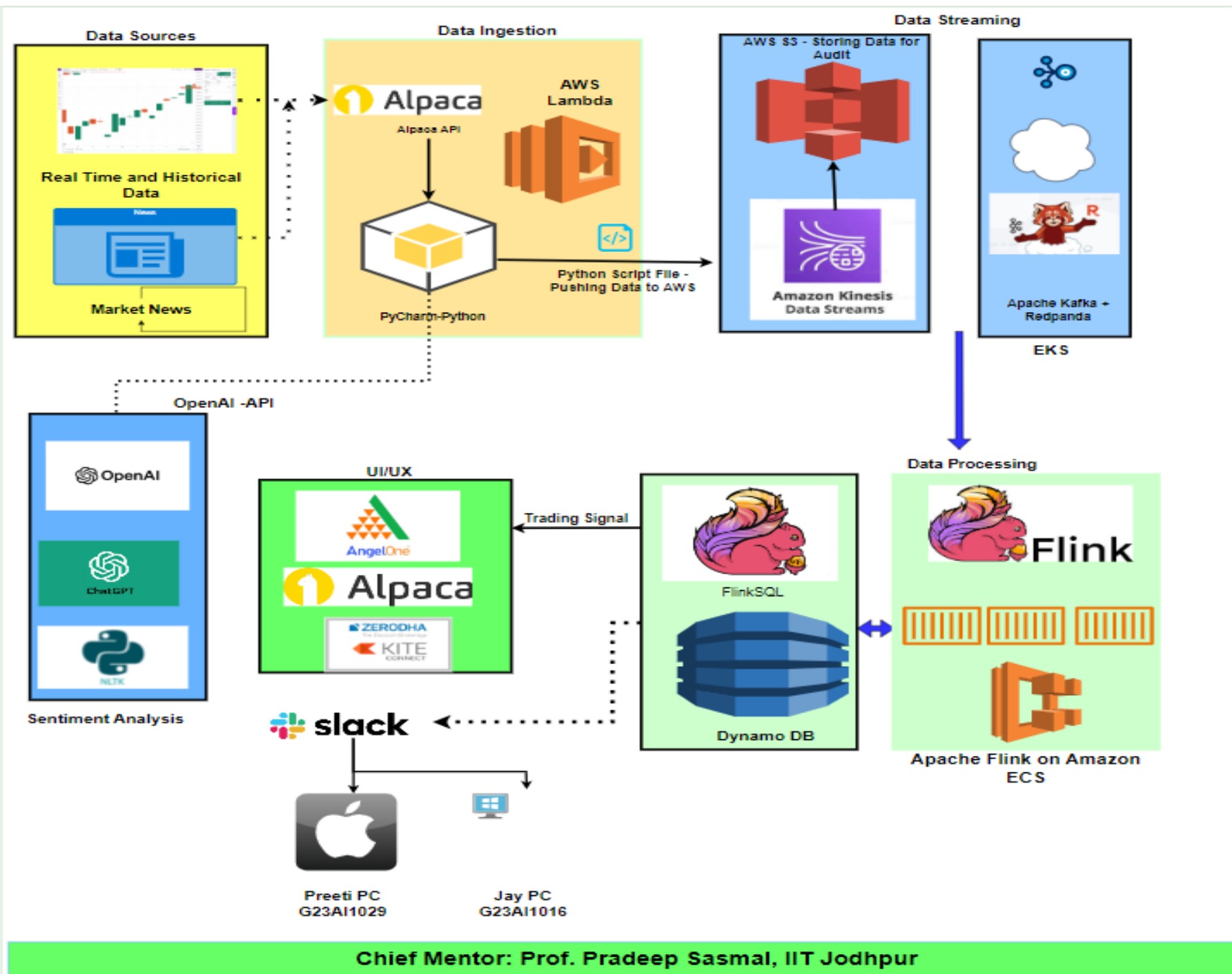


Project Objectives

The primary objectives of this project are:

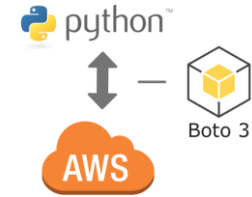
- **Data Collection:** Implement Python scripts to fetch historical and real-time stock prices and news data from APIs (e.g., Alpaca API, Market News APIs).
- **Real-Time Data Processing:** Utilize Apache Flink deployed on Amazon ECS to process streaming data from Amazon Kinesis Data Streams and generate 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



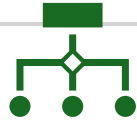
1. Data Sources

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:

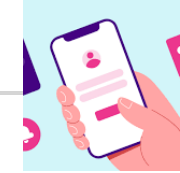
Processes sentiment analysis results to generate buy/sell signals based on real-time and historical data.



7. Trade Execution

Alpaca API:

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

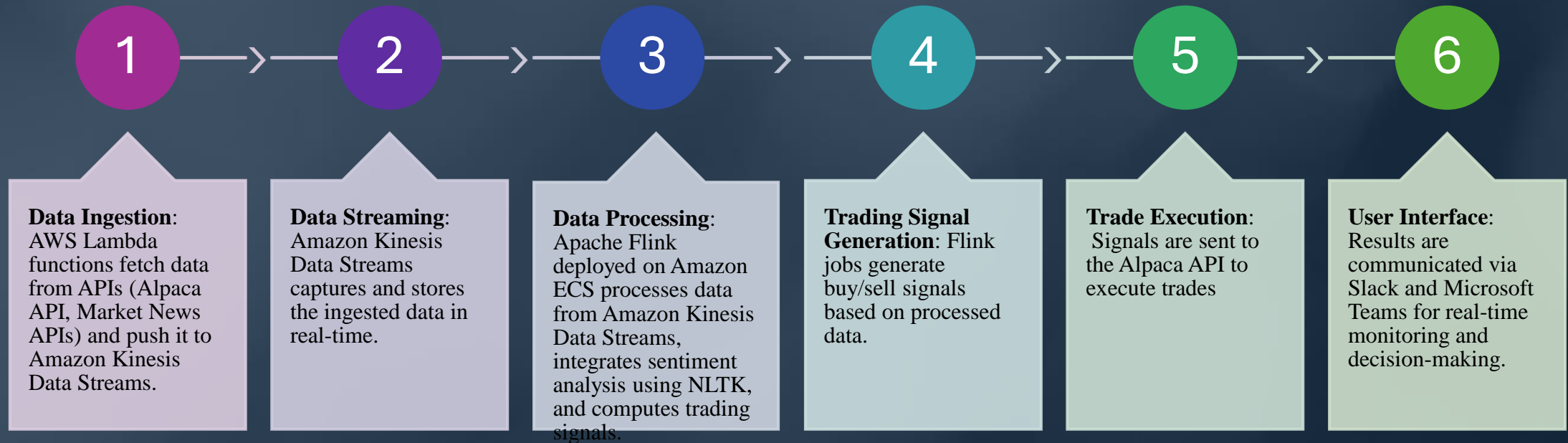


8. Data Processing

Slack/Microsoft Teams:

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

Data Flow



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 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.

Redpanda

Overview

Topics

Schema Registry

Consumer Groups

Security

Quotas

Connectors

Reassign Partitions

Cluster > Topics

Refreshing in 9 secs

3

3

Total Topics

Total Partitions

Create Topic

☐ Show internal topics

Name	Partitions	Replicas	CleanupPolicy	Size
market-news	1	1	delete	2.28 MiB
stock-prices	1	1	delete	21.2 MiB
trading-signals	1	1	delete	1.09 kiB

Total 3 items

< 1 >

50 / page



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

Available Task Slots

18

Total Task Slots 20 | Task Managers 1

Running Jobs

2

Finished 0 | Canceled 0 | Failed 0

Running Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
collect	2024-07-16 17:51:10.021	3d 21h 28m 30s	–	6 6	RUNNING
insert-into_default_catalog.default_database.trading_signals_sink	2024-07-16 17:50:52.389	3d 21h 28m 48s	–	6 6	RUNNING

Completed Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
No 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		

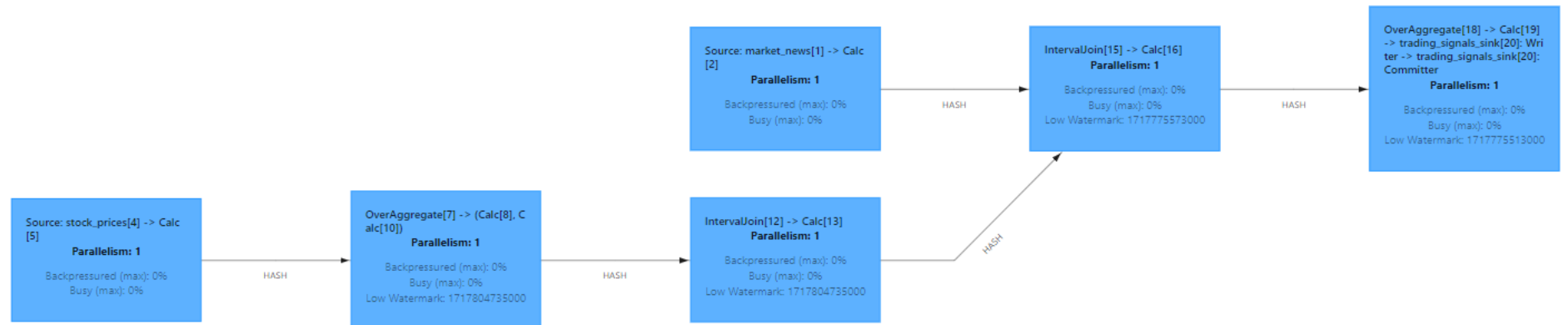
Overview

Exceptions

TimeLine

Checkpoints

Configuration



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.

ApacheFlink_GenAI_Trading

Version control

signal_handler.pyutils.pyddl.sqlnews-producer.pydocker-compose.ymlkeys.pyprices-producer.py

ApacheFlink_GenAI_Trading C:\Users\JayVis...
venv library root
alpaca_config
keys.py
libs
flink-sql-connector-kafka-3.1.0-1.18.jar
ddl.sql
docker-compose.yml
docker-entrypoint.sh

No data sources are configured to run this SQL and provide advanced code assistance.
SQL dialect is not configured. ClickHouse matches best.

CREATE OR REPLACE TABLE trading_signals_sink (
symbol STRING,
signal_time TIMESTAMP_LTZ,
signal STRING
) WITH (
'connector' = 'kafka',
'topic' = 'trading-signals',

LocalLocal (3)

SQL Query Result (Table)
Page: 52 of 53
Updated: 09:52:48.747

symbol	headline	sentiment	close	sma_20	sma_50	news_time	price_time
AAPL	DexCom Announced Today That T~	0	194.72	194.82295	194.82295	2024-06-05 10:03:09.000	2024-06-05 10:03:00.000
AAPL	Nvidia's AI Muscle Propels St~	0	194.95	194.81375	194.81375	2024-06-05 10:39:43.000	2024-06-05 10:39:00.000
AAPL	Market Clubhouse Morning Memo~	0	195.12	195.33505	195.33505	2024-06-05 14:23:29.000	2024-06-05 14:23:00.000
AAPL	Market Clubhouse Morning Memo~	0	195.136	195.32817	195.32817	2024-06-05 14:23:29.000	2024-06-05 14:24:00.000
AAPL	Benzinga Market Summary: Bitc~	0	195.2819	195.36305	195.36305	2024-06-05 14:31:40.000	2024-06-05 14:31:00.000
AAPL	Benzinga Market Summary: Bitc~	0	195.405	195.36554	195.36554	2024-06-05 14:31:40.000	2024-06-05 14:32:00.000
AAPL	CNBC Halftime Report Final Tr~	0	196.045	196.07838	196.07838	2024-06-05 17:00:25.000	2024-06-05 17:00:00.000
AAPL	CNBC Halftime Report Final Tr~	0	195.96	196.07329	196.07329	2024-06-05 17:00:25.000	2024-06-05 17:01:00.000
AAPL	S&P 500, Nasdaq 100 Jointly H~	1	195.55	195.862	195.862	2024-06-05 20:18:04.000	2024-06-05 20:18:00.000
AAPL	S&P 500, Nasdaq 100 Jointly H~	1	195.55	195.85019	195.85019	2024-06-05 20:18:04.000	2024-06-05 20:19:00.000
AAPL	Nvidia Crosses \$3T Market Cap~	1	195.61	195.58414	195.58414	2024-06-05 20:38:07.000	2024-06-05 20:38:00.000
AAPL	Nvidia Crosses \$3T Market Cap~	1	195.65	195.57994	195.57994	2024-06-05 20:38:07.000	2024-06-05 20:39:00.000
AAPL	Dan Niles Predicts 'So Much M~	0	195.93	195.61671	195.61671	2024-06-06 08:08:56.000	2024-06-06 08:08:00.000
AAPL	Dan Niles Predicts 'So Much M~	0	195.98	195.62456	195.62456	2024-06-06 08:08:56.000	2024-06-06 08:09:00.000
AAPL	The Competition Authority Con~	1	195.87	195.80719	195.80719	2024-06-06 13:04:39.000	2024-06-06 13:04:00.000
AAPL	The Competition Authority Con~	1	195.85	195.80219	195.80219	2024-06-06 13:04:39.000	2024-06-06 13:05:00.000
AAPL	Apple, Waste Connections And ~	0	195.87	195.80399	195.80399	2024-06-06 13:06:28.000	2024-06-06 13:06:00.000
AAPL	Apple, Waste Connections And ~	0	195.81	195.80362	195.80362	2024-06-06 13:06:28.000	2024-06-06 13:07:00.000
AAPL	Google Accelerates AI Innovat~	0	195.93	195.8092	195.8092	2024-06-06 13:14:26.000	2024-06-06 13:14:00.000
AAPL	Google Accelerates AI Innovat~	0	195.88	195.80959	195.80959	2024-06-06 13:14:26.000	2024-06-06 13:15:00.000
AAPL	Market Clubhouse Morning Memo~	0	195.84	195.7631	195.7631	2024-06-06 14:10:40.000	2024-06-06 14:10:00.000
AAPL	Market Clubhouse Morning Memo~	0	195.96	195.76633	195.76633	2024-06-06 14:10:40.000	2024-06-06 14:11:00.000
AAPL	Nvidia Surpasses Apple On AI ~	-1	195.6013	195.72893	195.72893	2024-06-06 15:43:36.000	2024-06-06 15:43:00.000
AAPL	Nvidia Surpasses Apple On AI ~	-1	195.61	195.72655	195.72655	2024-06-06 15:43:36.000	2024-06-06 15:44:00.000
AAPL	Apple To Unveil New Password ~	0	194.46	194.53725	194.53725	2024-06-06 20:29:33.000	2024-06-06 20:29:00.000

Inc Refresh
Dec Refresh

Goto Page
Last Page

Next Page
Prev Page

Open Row

PARTITION

START OFFSET

MAX RESULTS

FILTER

All

Newest-50

50

☐



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.

←

→

↺

🔍 app.alpaca.markets/user/profile#:tab

☆

⬇

👤

🐎

Live

553186809

⬆

🏠

Home

📁

Account

>

☁

Alpaca Connect

♥

Plans & Features

>

💰

Funds & Wallet

>

📄

API

💬

Community

>

🛠

Support

⚖

Legal

Crypto Fee Rate

Level 1

🔍

Search

Get Started with Options

+ Add Funds

Your Profile

Basic Info

Additional Info

Trusted Contact

Agreements

Security

Crypto Fees

IP Allowlist

Details

Account Number

553186809

Full Name

Jay Vishvakarma

Address

331, Naidu Colony Pant Nagar Ghat Kopar

Update Details

Email

jay.vishvakarma@outlook.com

Phone Number

🇮🇳 +91 88176-59208

Save changes

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

PK-H98QULS4K8DXYO5DUJ

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

