|  |
| --- |
| TECHNICAL REPORT |

|  |
| --- |
| Electrical & Computer Engineering & Computer Science (ECECS) |

|  |  |
| --- | --- |
| 2nd SEMESTER |  |



Contents

**Project Name …………………………….2**

**Executive Summary …………………….2**

**Technical Report …………………………3**

**Highlights of Project ……………………..3**

**Submitted on: …………………………….3**

**Abstract ……………………………………4**

**Methodology ……………………………....5**

**Data Pipeline ……………………………...7**

**Execution ………………………………….8**

**Results Section …………………………...13**

**Conclusion …………………………………21**

**Contributions/References ………………22**

|  |  |  |
| --- | --- | --- |
| Executive Summary person at a table writing in a notebook with people around | | |
|  | | |
| Team Members:   * SIRISHA KASETTY * VENKATESH SWARNA * MAMATHA PEDDAMALA * LIKITH KAGITA | Questions?  Contact:  [Skase1@unh.newhaven.edu](mailto:Skase1@unh.newhaven.edu)  [vswar1@unh.newhaven.edu](mailto:vswar1@unh.newhaven.edu)  [mpedd3@unh.newhaven.edu](mailto:mpedd3@unh.newhaven.edu)  [lkagi1@unh.newhaven.edu](mailto:lkagi1@unh.newhaven.edu) |  |

|  |
| --- |
| Uber & Lyft Data Analysis using AWS |

|  |  |
| --- | --- |
| STOCK MARKET REAL TIME DATA ANALYSIS USING KAFKA |  |
| Highlights of Project  * Able to fetch the real time data from data feeds using Beautiful Soup and Python SDk * Transfer the data through Apache Kafka(Data Streaming) platform. * Uploaded the data into S3 bucket received at the Apache Kafka Consumer end * Created DB using AWS Glue and set up AWS Crawler to crawl S3 Bucket * Using Simba ODBC connector, connected to PowerBI desktop to run the visualizations  Submitted on: 04/22/2024. |

## Abstract

The volatility and complexity of the stock market demand advanced solutions for real-time data analysis to drive informed decision-making. This project leverages Apache Kafka, a robust distributed messaging system, to develop a scalable infrastructure capable of handling massive volumes of high-velocity stock market data. Our system integrates diverse data sources, including financial APIs, news feeds, and economic indicators, providing a holistic view of market dynamics. The architecture supports real-time data processing, cleaning, transformation, and enrichment to identify trends, predict stock movements, and detect trading anomalies. We employ a meticulously designed data pipeline that ensures data integrity and reduces latency, enabling the delivery of timely and actionable insights. The success of the project is measured through key performance metrics such as system scalability, processing latency, and data accuracy. This initiative not only enhances our understanding of market behaviors but also serves as a critical tool for traders and financial analysts aiming to capitalize on market opportunities efficiently.

## 

## Crisp-DM Methodology

The CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology is a widely used framework for guiding data mining and analytics projects.

CRISP-DM phases:

* Business Understanding
* Data Understanding
* Data Preparation
* Modeling
* Evaluation

1. Business Understanding: This phase involves defining the objectives and requirements of the project from a business perspective. For a stock market data analysis project, this would mean identifying the key performance indicators (KPIs) such as predicting stock movements, identifying market trends, and detecting anomalies in trading patterns. It also includes setting the goals for latency, accuracy, and scalability of the system.
2. Understanding Data: This stage focuses on collecting initial data and proceeding with activities to get familiar with the data, identify data quality issues, discover first insights into the data, or detect interesting subsets to form hypotheses for hidden information. For the stock market analysis, this involves understanding the types of data available from financial APIs, news feeds, and economic indicators, and how they can be correlated to influence stock prices.
3. Data Preparation: The data preparation phase covers all activities needed to construct the final dataset from the initial raw data. This may include table, record, and attribute selection, as well as data cleaning and construction of new attributes. In your project, this could involve setting up Kafka to ingest and manage streams of data, handling missing values and outliers, and integrating and aligning data from various sources into a consistent format.
4. Modeling: In this phase, various modeling techniques are selected and applied to prepare the models based on the data prepared in the previous stage. For analyzing the stock market, machine learning algorithms or statistical methods to forecast stock prices or identify patterns could be applied. This stage would also involve tuning parameters and choosing the right algorithms that work best with time-series financial data.
5. Evaluation: After modeling, this stage assesses the model or models to ensure they meet the business objectives set in the first phase. Your project would evaluate the latency of the data pipeline, the accuracy of the predictions, and the scalability of the Kafka setup. It's crucial to review whether the results truly answer the project’s initial questions and objectives.
6. Deployment: The final phase involves deploying the data mining solution to the business. This could mean implementing the Kafka-based system in a live environment where it can process real-time stock market data and provide insights through dashboards, alerts, or automated trading systems. It also includes setting up the system for ongoing monitoring and maintenance to ensure it adapts to new data or changes in market conditions.

## Data Pipeline

## A diagram of a data storage system Description automatically generated

**Data Engineering Pipelin**e Schema

* + Data Ingestion: Data is collected using SDK boto3, which suggests that the AWS SDK for Python is used to interact with Amazon Web Services. Beautiful Soup is also mentioned, indicating that web scraping is utilized to gather data, likely from web pages.
  + Data Storage: S3, or Simple Storage Service, is an AWS service used for object storage. It's highly scalable and used here to store the streaming data processed by Kafka.
  + Data Processing: A Crawler, likely part of AWS Glue, is used to catalog and organize the data within S3. AWS Glue Data Catalog is a central repository to store structural and operational metadata for all the data assets. Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL.
  + Data Consumption: Power BI, a business analytics tool by Microsoft, is used to visualize and analyze the data. The Simba ODBC connector is likely used to connect Power BI to Athena or another database, allowing for SQL queries and data retrieval for reporting.
  + Data Streaming : Apache Kafka is the streaming platform that takes in the ingested data. Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, and wicked fast.

**Create EC2 instance. After that follow below steps.**

**Create AWS Credentials folder to access its resources further in code**

In SSH; go to home if not already there (cd ~)

1. Create new folder:

mkdir .aws

2. Create a new file

vim .aws/credentials

3. paste entire AWS CLI content – as it is – save and quit

[you need to modify the credentials file on every launch since AWS CLI is not constant]

**To Install Jupyter and run on EC2**

**SSH into the instance and type below commands one by one.**

1. sudo su (root user)

2. yum update (to perform updates, of any required)

3. Create and Activate python virtual env

python3 -m venv venv (To create virtual environment) source venv/bin/activate (To activate virtual environment)

4. Install the required packages – make note that there were no errors [warnings can be ignored]

pip install pyyaml ipython jupyter ipyparallel pandas boto3 -U

5. Enable IPython Cluster

ipcluster nbextension enable

6. Start an ipcluster with 4 engines

ipcluster start -n 4

[DO NOT stop/interrupt the process – otherwise you will be unable to finish this assignment]

7. Let the previous SSH terminal be as it is and start a new session (SSH again to the instance)

source venv/bin/activate

jupyter notebook --port=8888 --no-browser --ip=0.0.0.0 --allow-root

**Commands to Download Kafka**

wget https://downloads.apache.org/kafka/3.7.0/kafka\_2.13-3.7.0.tgz

tar -xvf kafka\_2.13-3.7.0.tgz

**Java is Prerequisite to run Kafka.**

java -version (to check java version)

sudo yum install java-1.8.0-openjdk (to install java)

**Change to Kafka directory before starting server**

cd kafka\_2.13-3.7.0

**To Start Zoo-keeper:**

bin/zookeeper-server-start.sh config/zookeeper.properties

After running Zoo-keeper, open new terminal and SSH into it to start the Kafka Server

**Start Kafka-server:**

export KAFKA\_HEAP\_OPTS="-Xmx256M -Xms128M"

cd kafka\_2.13-3.7.0

By default, host pointed to the private address. So update server.properties to run it on public IP

To do this, Do "sudo nano config/server.properties" - change ADVERTISED\_LISTENERS to public ip of the EC2 instance. After that run below command to run the Kafka Server with updated config file.

bin/kafka-server-start.sh config/server.properties.

After running Kafka-Server, again open new terminal and SSH.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Producer to Consumer data transition

A screenshot of a computer

Description automatically generated

Topic Creation

cd kafka\_2.13-3.7.0

bin/kafka-topics.sh --create --topic demotest --bootstrap-server 52.91.127.14:9092 --replication-factor 1 --partitions 1 (Here we are creating topic name called demotest. You can give desire name)

A screen shot of a computer

Description automatically generated

To get the data from NASDAQ Stock Exchange

A screenshot of a computer

Description automatically generated

Kafka Producer code through sending data

bin/kafka-console-producer.sh --topic demotest --bootstrap-server 52.91.127.14:9092

After running Kafka-Server, again open new terminal and SSH.

A screenshot of a computer

Description automatically generated

Kafka Consumer

cd kafka\_2.13-3.7.0

bin/kafka-console-consumer.sh --topic demotest --bootstrap-server 52.91.127.14:9092

write anything in producer and that text you can find in consumer. Great your producer and consumer are ready.

A screenshot of a computer

Description automatically generated

Pushing Data to S3 Bucket

A screenshot of a computer

Description automatically generated

## 

**Kafka\_Producer.ipynb**

**import** pandas **as** pd

**from** kafka **import** KafkaProducer

**import** time

**from** json **import** dumps

**import** json

**from** stockPriseGetter **import** stockGetPrice

producer = KafkaProducer(bootstrap\_servers=['54.144.131.204:9092'],

value\_serializer=lambda x:

dumps(x).encode('utf-8'))

**while** **True**:

stockGetPriceObject **=** stockGetPrice()

stockPrices **=** stockGetPriceObject**.**getPrice()

*# Send the entire stockPrices object as a single message*

producer**.**send('demotest', value**=**stockPrices)

time**.**sleep(5)

print('After Sleep')

## **stockPriseGetter.py**

import requests

from bs4 import BeautifulSoup

from datetime import datetime

class stockGetPrice:

stocks\_dict = {

"Apple": "AAPL",

"Microsoft Corp": "MSFT",

"Tesla Inc": "TSLA"

}

@staticmethod

def getStockPrice(index):

url = f'https://www.google.com/finance/quote/{index}:NASDAQ'

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

div\_price\_class = "YMlKec fxKbKc"

price = float(soup.find(class\_=div\_price\_class).text.strip()[1:].replace(",", ""))

return price

def getPrice(self):

prices = {}

for name, value in self.stocks\_dict.items():

price = self.getStockPrice(value)

prices[name] = price

# Convert datetime object to string

current\_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

prices['date'] = current\_time

return prices

## Results Section

S3 Bucket Data

A screenshot of a computer

Description automatically generated

Data Processing consumer logs

A screen shot of a computer

Description automatically generated

Simba Connector to connect Athena to PowerBI

A screenshot of a computer

Description automatically generated

PowerBI Report

A screenshot of a computer

Description automatically generated

## Conclusion

The project's architecture establishes a resilient and scalable solution for real-time stock market data analysis. Through the utilization of Apache Kafka, we efficiently managed high-velocity data streams, ensuring a robust data ingestion process. AWS services such as Boto3, S3, Glue, and Athena provided a seamless data storage and processing experience, capable of handling the vast volume and variety of stock market data. The employment of web scraping technologies like Beautiful Soup enabled the incorporation of diverse datasets, enhancing the richness of the analysis.

The integration of AWS Glue Crawler and Athena facilitated a structured and query-able data environment, allowing for sophisticated data processing techniques to be applied. The data storage solution, anchored by S3, provided the necessary durability and accessibility to support the real-time nature of stock market fluctuations.

Finally, the deployment of Power BI, connected through the Simba ODBC connector, empowered end-users with insightful visualizations and interactive analytics tools. The system's success was evidenced by the precise and rapid insights it provided, allowing for timely decision-making in a domain where seconds can equate to significant financial implications. This architecture not only addressed the initial challenges posed by the dynamic nature of the stock market but also laid a foundation for continuous improvement and adaptability in the face of evolving data analytics needs.

In conclusion, the project's strategic use of streaming data platforms, cloud storage, and visualization tools underpinned a successful approach to real-time data analytics in the stock market domain, providing a competitive edge in financial analysis and trading strategies.

## Contributions/References

* Apache Kafka Documentation: Provided essential guidelines for setting up and managing the Kafka streaming service.
* AWS Official Documentation: Served as the primary reference for utilizing Boto3, AWS Glue, S3,
* and Athena services for data ingestion, storage, and processing workflows.
* Beautiful Soup Documentation: Informed strategies for effective web scraping, crucial for enriching our data sources.
* Microsoft Power BI Documentation: Guided the creation of interactive dashboards for data consumption and business intelligence.
* Simba Technologies: Information on ODBC connectors for seamless integration between Athena and Power BI.
* R. Hecht-Nielsen, “Confabulation Theory: The Mechanism of Thought,” for foundational concepts in data analysis algorithms.
* L. Breiman, “Random Forests,” Machine Learning, 2001, for the machine learning techniques applied in data analysis.
* L. Torgo, “Data Mining with R, learning with case studies,” Chapman and Hall/CRC, for guidance on using R for data analysis, which informed our data processing strategies even though we implemented them in a different environment.
* J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” for understanding scalable data processing patterns which we adapted for our streaming data model.