Final Project Report

on

Stock Market Data Streaming

by:

Shail Patel (CWID: 885175489)

Malay Koladiya (CWID: 885867812)

CPSC 531-01 13480
Advanced Database Management

Spring 2023

Prof. Tseng-Ching James Shen

Department of Computer Science

California State University, Fullerton

Spring 2023

Contents

Introduction

In recent years, the subject of the stock market has increasingly become popular for traders, analysts, and investors. With the advent of technology, stock market data streaming has emerged as an essential tool for people who wish to keep up with the most recent developments and market trends. Stock market data streaming refers to the real-time delivery of stock market data, which includes information such as the stock's trading value, stock price, and other financial indicators.

The availability of real-time data has completely changed how stock market traders and investors operate. Trading professionals can make well-informed decisions regarding buying and selling stocks by accessing the latest information. The availability of real-time data is especially crucial in the constantly-evolving and fast-paced world of the stock market.

As part of this project, we will explore stock market data streaming. We will use different technologies to create a system that offers insightful information on the stock market. Using our technology, users can follow market trends, evaluate stock performance, and make wise investment decisions in real-time.

Technologies and Tools Used

Real-time data is essential for making wise investment decisions in the dynamics and complicated stock market environment. Traditional data processing methods are no longer adequate due to the growing volume of the data. Therefore, we will use technologies like AWS EC2, Kafka, Zookeeper, Databricks, and PySpark.

1. AWS EC2
    a. Amazon Web Service EC2 is used to host the Kafka cluster. AWS EC2 is a web service that provides resizable computing capacity in the Cloud
2. Kafka
    a. Kafka is a distributed streaming platform for building real-time data pipelines and streaming applications.
3. Zookeeper
    a. Zookeeper is a distributed coordination service often used in distributed systems for managing configuration information and synchronization.
    b. In this project, Zookeeper is used to manage the Kafka cluster. It will help manage and synchronize the configuration of Kafka brokers, producers, and consumers.
4. Databricks
    a. Databricks is a cloud-based data analytics platform providing a unified data processing and analytics approach.
    b. It is designed to handle large-scale data processing and analytics using Apache-Spark.
5. Pyspark
    a. Pyspark is an API in Python for Spark. Pyspark provides an interface for streaming data processing and analysis, allowing real-time data manipulation.
    b. This project uses Pyspark to stream structured data from Kafka consumers using the Spark Structured Streaming library. We will be using built-in features of Pyspark, such as SQL queries.

Dataset

In this project, we are gathering data from two different sources. One would be through yahoo finance using the finance library available in Python, and the other through the dataset available on Kaggle. The ‘yfinance’ library is a Python package that provides a convenient interface to retrieve historical market data and financial information from Yahoo Finance. It allows you to easily access and download data such as historical stock prices, dividend data, company information, and more.

Functionalities

In this project, we will use Kafka, Databricks, and AWS to build a reliable system for streaming stock market data.

The application will read the data from two sources. The first source is Yahoo Finance, and the second is CSV files collected from kagle.com

The data is then analyzed using Databricks. In the Databricks notebook, we created different trading strategies to analyze the data. Based on the analysis, the trading strategies will signal the user when to buy, sell, or hold the stock. Based on the data source, we added different columns to the dataset, such as RSI, EMA, SMA, and Signal columns.

After analyzing and updating the dataset, the dataset's sample rows are extracted and sent to the Kafka producer. The Kafka producer then writes the data to the Kafka consumer through Kafka topics. The Kafka cluster is running on an AWS EC2 instance.

Upon receiving the data on the Kafka consumer, we will leverage the Spark Structured Streaming library to stream the data in real time. Finally, the streaming data will be available on the dashboards, where users can visualize and interpret the data.

What is SMA?

The most commonly used moving average is a so-called simple moving average (SMA), which is the average closing price of a given security over a specific number of days. For example, you can find a stock's 20-day SMA by adding its prices over 20 days, then dividing that number by 20.

What is EMA?

Exponential Moving Average (EMA) is similar to Simple Moving Average (SMA), measuring trend direction over a period of time. However, whereas SMA simply calculates an average of price data, EMA applies more weight to more current data.

What is MACD?

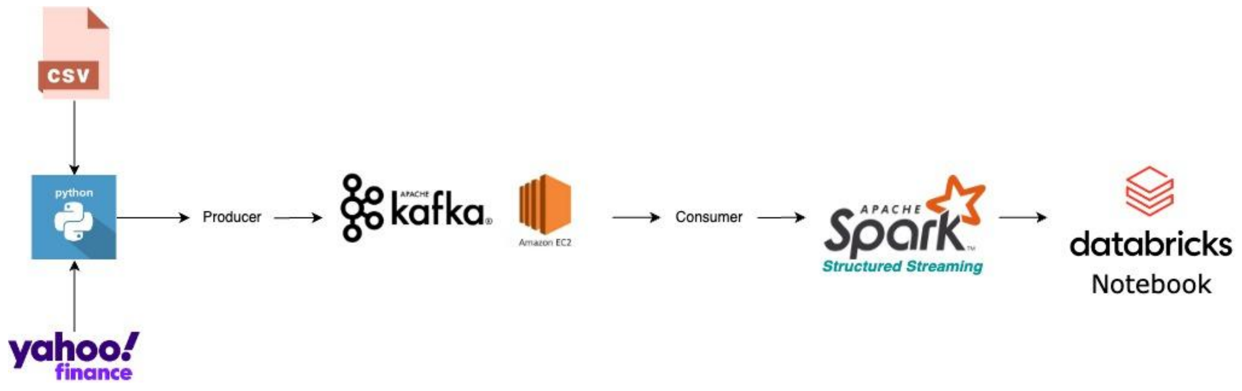
The Moving Average, Convergence/Divergence indicator, is a momentum oscillator primarily used to trade trends. Although it is an oscillator, it is not typically used to identify overbought or oversold conditions. It appears on the chart as two lines that oscillate without boundaries.

What is RSI?

Description. The Relative Strength Index (RSI), developed by J. Welles Wilder, is a momentum oscillator that measures the speed and change of price movements. The RSI oscillates between zero and 100. Traditionally the RSI is considered overbought when above 70 and oversold when below 30.

Architecture & Design

The following diagram shows an overview of the Architecture and Design of the project.



The data gathered is analyzed and sent to the Kafka cluster hosted on an Amazon Web Service EC2 instance. The Kafka producer then writes the data through topics and sends it to the Kafka consumer. After receiving the data on the Kafka consumer, it is then streamed through the Spark Structured Streaming library. This streamed data can be viewed on the Dashboard on Databricks.

GitHub Link

The files of the project can be accessed on GitHub through the following link:

https://github.com/shail2811/stock_market_data_streaming

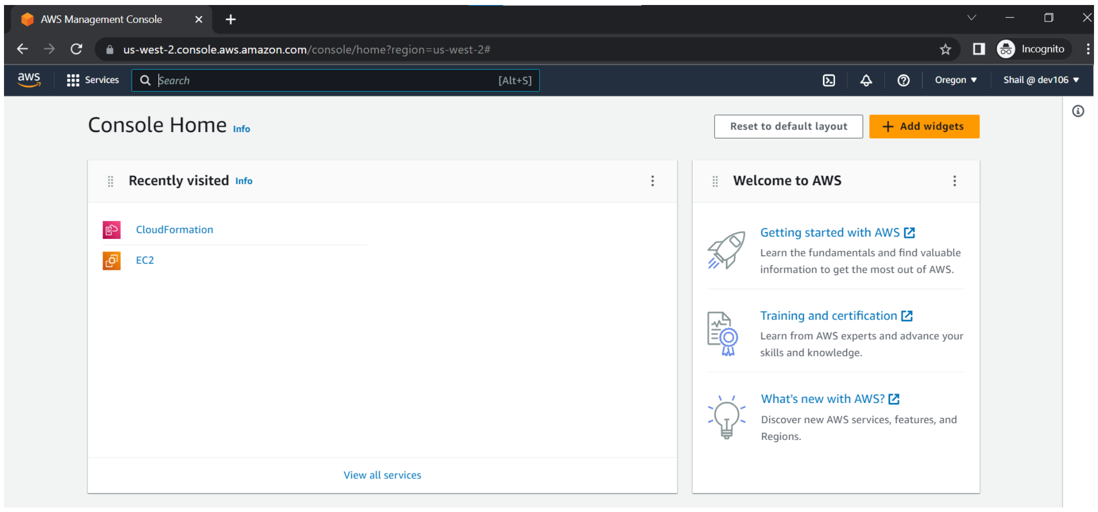
Deployment Instructions

Step 1: Login into the AWS account with the proper username and password.

Step 2: Go to this link to log in - https://dev106.signin.aws.amazon.com/console

And you will be able to see the following screen after successful login

Step 3: Click on EC2 as seen on the screen

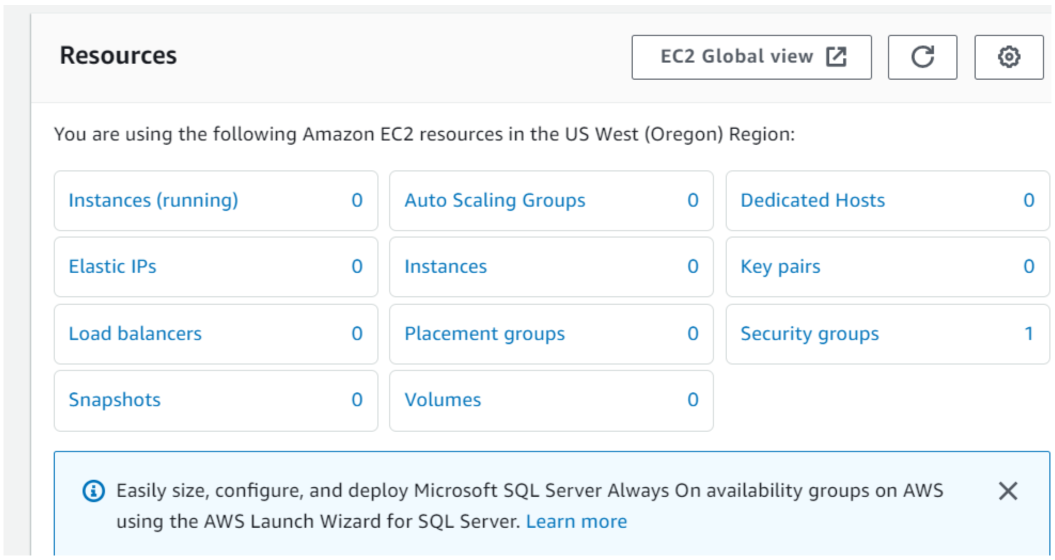


Learn the fundamentals and find valuable information to get the most out of AWS.

Learn from AWS experts and advance your skills and knowledge.

Discover new AWS services, features, and Regions.

Step 4: Click on Instances

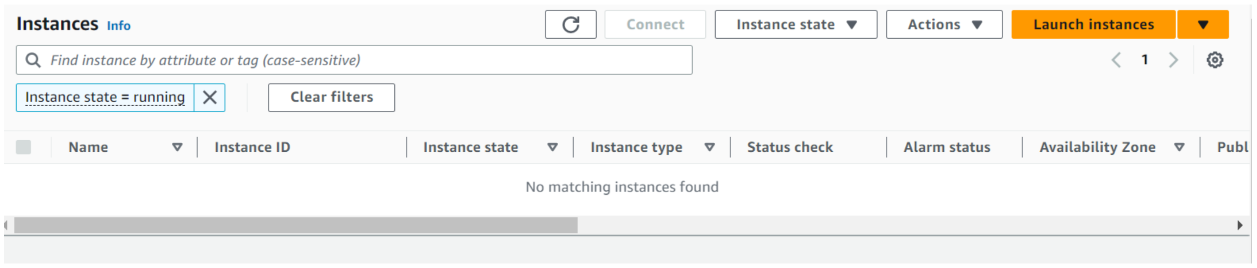


You are using the following Amazon EC2 resources in the US West (Oregon) Region:

Instances (running)	0	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	0	Instances	0	Key pairs	0
Load balancers	0	Placement groups	0	Security groups	1
Snapshots	0	Volumes	0		

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

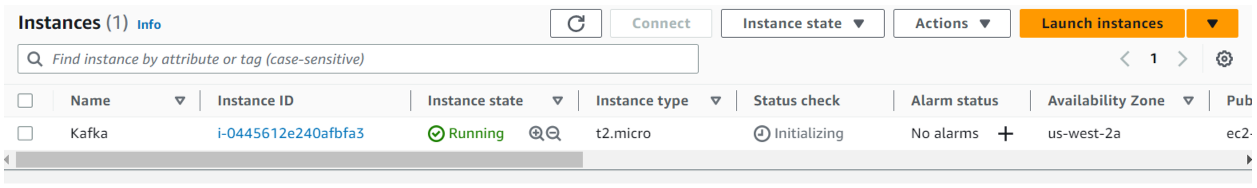
Step 5: Click on the launch instance



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publ

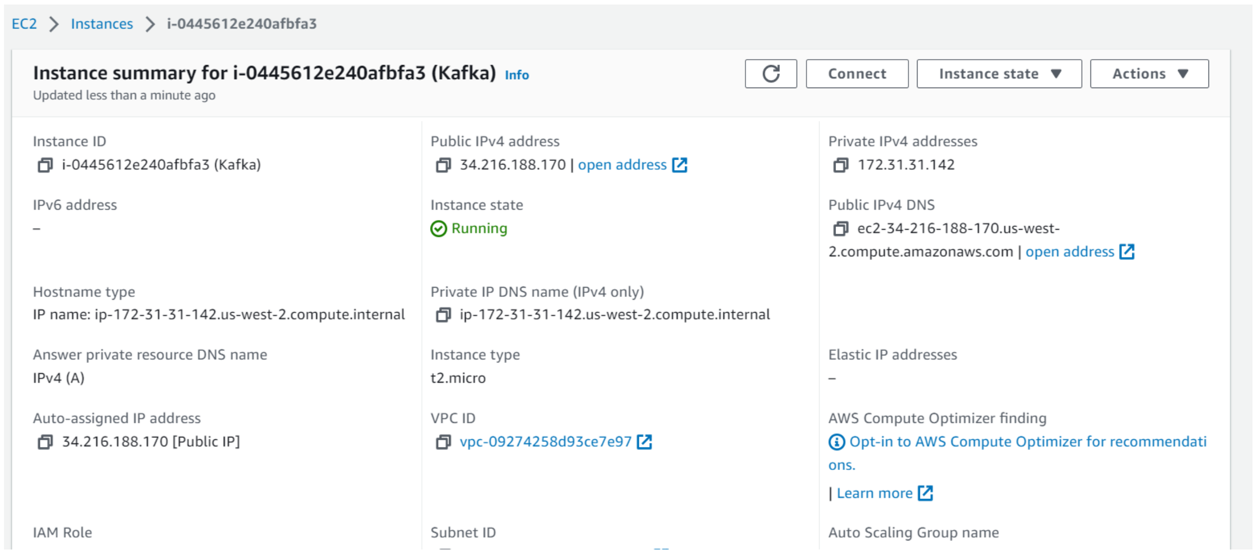
No matching instances found

Step 6: Give a name to your instance, create a keypair, keep the rest of the configurations to default, and click on launch instance.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pub
Kafka	i-0445612e240afbfa3	Running	t2.micro	Initializing	No alarms	us-west-2a	ec2-

Step 7: Click on the instance id
Click on Connect



Instance summary for i-0445612e240afbfa3 (Kafka) Info
Updated less than a minute ago

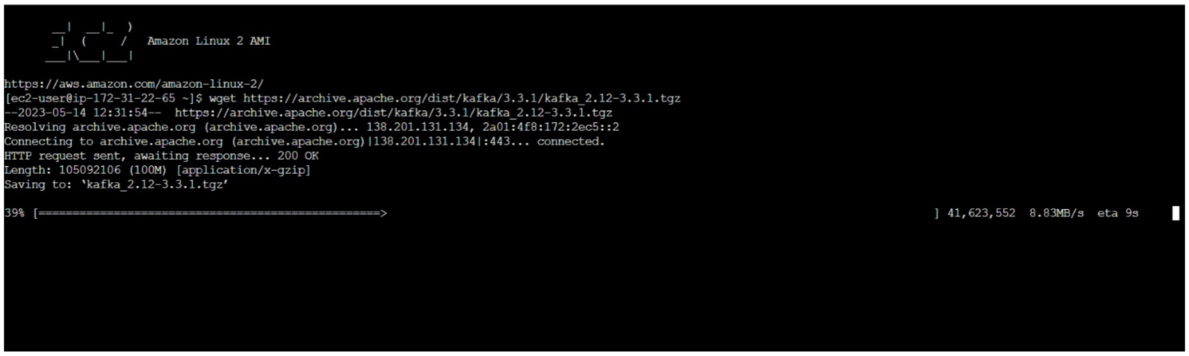
Instance ID i-0445612e240afbfa3 (Kafka)	Public IPv4 address 34.216.188.170 | open address	Private IPv4 addresses 172.31.31.142
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-34-216-188-170.us-west-2.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-31-142.us-west-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-31-142.us-west-2.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 34.216.188.170 [Public IP]	VPC ID vpc-09274258d93ce7e97	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role	Subnet ID	Auto Scaling Group name

Step 8: We will run the following commands on our instance to setup Kafka and Zookeeper

We get https://archive.apache.org/dist/kafka/3.3.1/kafka_2.12-3.3.1.tgz
tar -xvf kafka_2.12-3.3.1.tgz

java -version
sudo yum install java-1.8.0-openjdk
java -version
cd kafka_2.12-3.3.1

 # for Ubuntu -
sudo apt-get update
sudo apt-get install -y openjdk-8-jdk
java -version
cd kafka_2.12-3.3.1

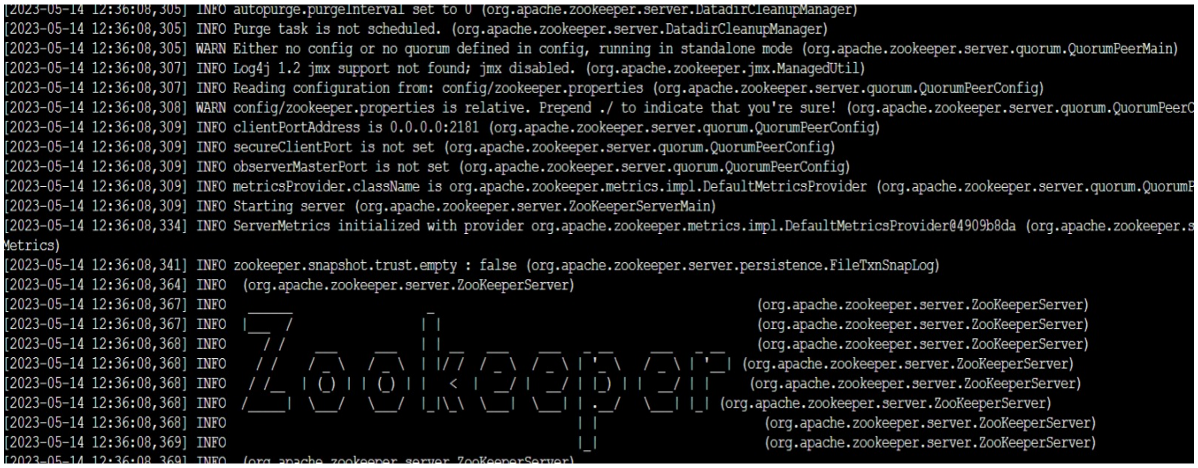


```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-22-65 ~]$ wget https://archive.apache.org/dist/kafka/3.3.1/kafka_2.12-3.3.1.tgz
--2023-05-14 12:31:54--  https://archive.apache.org/dist/kafka/3.3.1/kafka_2.12-3.3.1.tgz
Resolving archive.apache.org (archive.apache.org)... 138.201.131.134, 2a01:4f8:172:2ec5::2
Connecting to archive.apache.org (archive.apache.org)|138.201.131.134|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 105092106 (100M) [application/x-gzip]
Saving to: 'kafka_2.12-3.3.1.tgz'

39% [==================================================>                                                                    ] 41,623,552  8.83MB/s  eta 9s
```

Step 9: Start Zoo-keeper: bin/zookeeper-server-start.sh config/zookeeper.properties



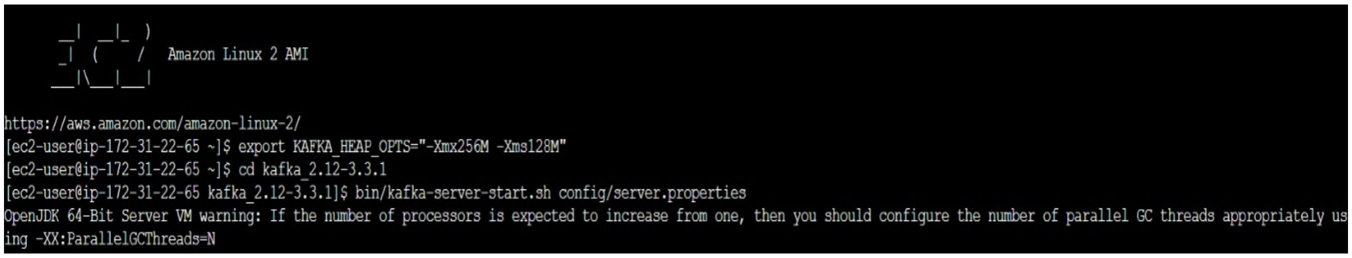
```
[2023-05-14 12:36:08,305] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-05-14 12:36:08,305] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-05-14 12:36:08,305] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2023-05-14 12:36:08,307] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2023-05-14 12:36:08,307] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-05-14 12:36:08,308] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerC
[2023-05-14 12:36:08,309] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-05-14 12:36:08,309] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-05-14 12:36:08,309] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-05-14 12:36:08,309] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumP
[2023-05-14 12:36:08,309] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2023-05-14 12:36:08,334] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@4909b8da (org.apache.zookeeper.s
Metrics)
[2023-05-14 12:36:08,341] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-05-14 12:36:08,364] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,367] INFO   ______                  _                                          (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,367] INFO  |___  /                 | |                                         (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,368] INFO     / /    ___     ___   | | __   ___    ___   _ __     ___   _ __   (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,368] INFO    / /    / _ \   / _ \  | |/ /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,368] INFO   / /__  | (_) | | (_) | |   <  |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,368] INFO  /_____|  \___/   \___/  |_|\_\  \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,368] INFO                                               | |                     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,369] INFO                                               |_|                     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-05-14 12:36:08,369] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
```

Step 10: Start Kafka-server:

Duplicate the session & enter in a new console --
export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
cd kafka_2.12-3.3.1
bin/kafka-server-start.sh config/server.properties

It is pointing to a private server. Change server.properties so that it can run on public IP

To do this, you can follow the approach shared below --
Do a "sudo nano config/server.properties" - change ADVERTISED_LISTENERS to public IP of the EC2 instance



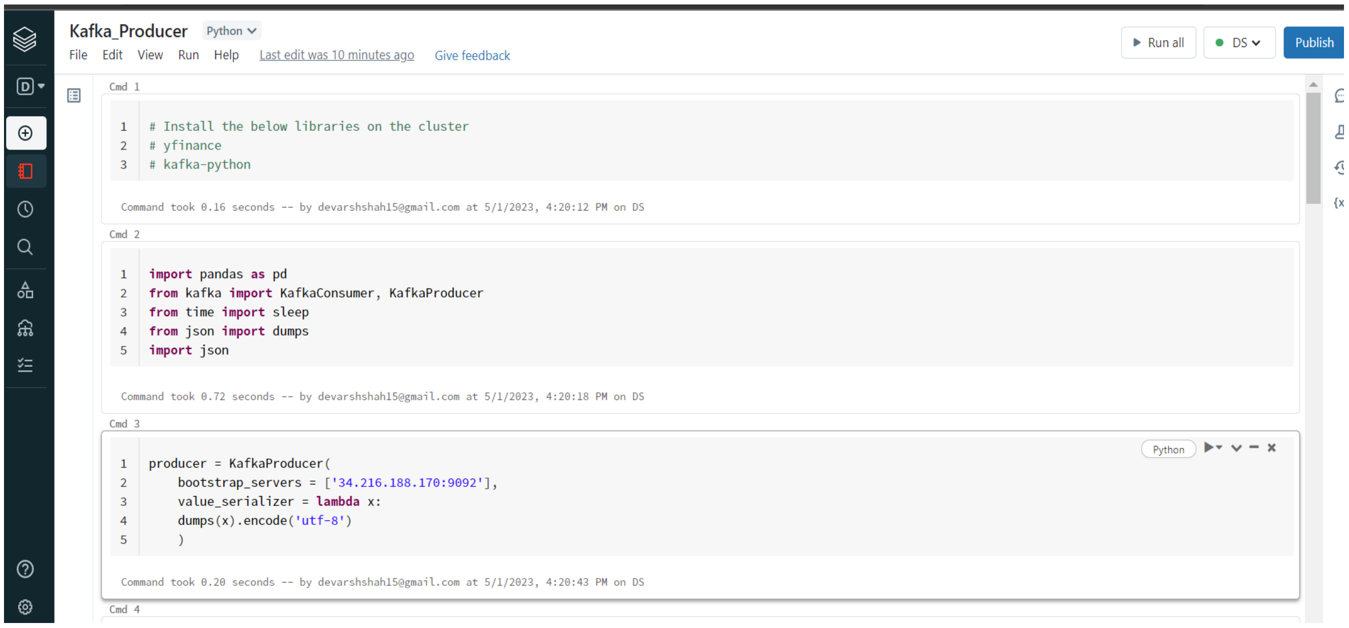
```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-22-65 ~]$ export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
[ec2-user@ip-172-31-22-65 ~]$ cd kafka_2.12-3.3.1
[ec2-user@ip-172-31-22-65 kafka_2.12-3.3.1]$ bin/kafka-server-start.sh config/server.properties
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should configure the number of parallel GC threads appropriately us
ing -XX:ParallelGCThreads=N
```

Step 11: Create the topic:

Duplicate the session & enter in a new console --
cd kafka_2.12-3.3.1
bin/kafka-topics.sh --create --topic demo_testing2 --bootstrap-server 54.215.179.39:9092 --replication-factor 1 --partitions 1

Step 12: Start Producer:
bin/kafka-console-producer.sh --topic demo_testing2 --bootstrap-server 54.215.179.39:9092



Cmd 1

```
# Install the below libraries on the cluster
# yfinance
# kafka-python
```

Command took 0.16 seconds -- by devarshshah15@gmail.com at 5/1/2023, 4:20:12 PM on DS

Cmd 2

```
import pandas as pd
from kafka import KafkaConsumer, KafkaProducer
from time import sleep
from json import dumps
import json
```

Command took 0.72 seconds -- by devarshshah15@gmail.com at 5/1/2023, 4:20:18 PM on DS

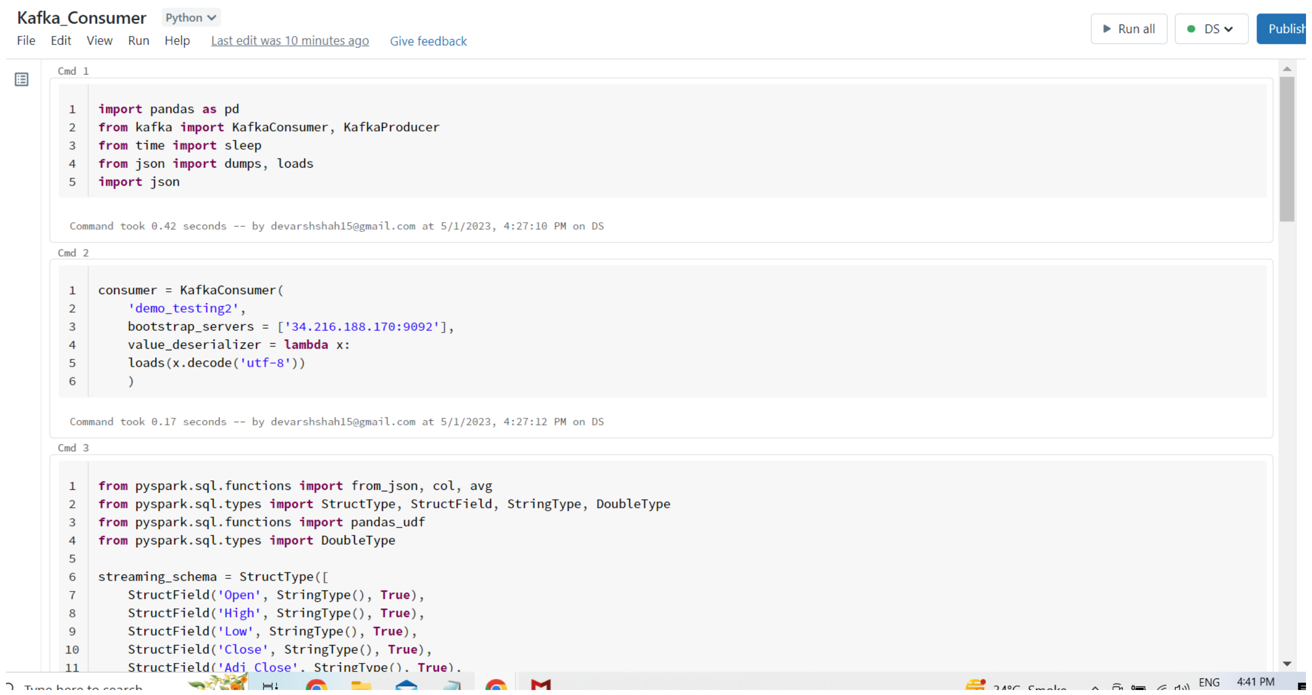
Cmd 3

```
producer = KafkaProducer(
    bootstrap_servers = ['34.216.188.170:9092'],
    value_serializer = lambda x:
    dumps(x).encode('utf-8')
    )
```

Command took 0.20 seconds -- by devarshshah15@gmail.com at 5/1/2023, 4:20:43 PM on DS

Cmd 4

Step 13: Start Consumer:
Duplicate the session & enter in a new console --
cd kafka_2.12-3.3.1
bin/kafka-console-consumer.sh --topic demo_testing2 --bootstrap-server 54.215.179.39:9092



Kafka_Consumer Python

File Edit View Run Help Last edit was 10 minutes ago Give feedback

Cmd 1

```
import pandas as pd
from kafka import KafkaConsumer, KafkaProducer
from time import sleep
from json import dumps, loads
import json
```

Command took 0.42 seconds -- by devarshshah15@gmail.com at 5/1/2023, 4:27:10 PM on DS

Cmd 2

```
consumer = KafkaConsumer(
    'demo_testing2',
    bootstrap_servers = ['34.216.188.170:9092'],
    value_deserializer = lambda x:
    loads(x.decode('utf-8'))
    )
```

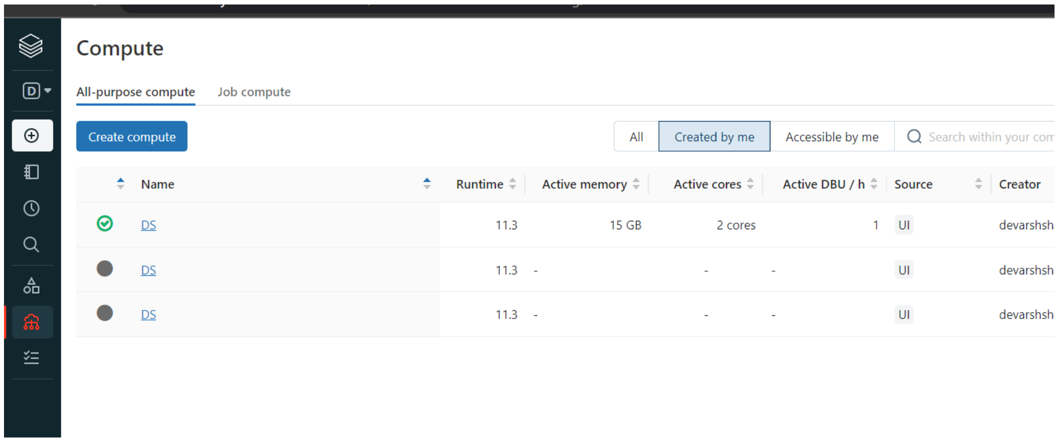
Command took 0.17 seconds -- by devarshshah15@gmail.com at 5/1/2023, 4:27:12 PM on DS

Cmd 3

```
from pyspark.sql.functions import from_json, col, avg
from pyspark.sql.types import StructType, StructField, StringType, DoubleType
from pyspark.sql.functions import pandas_udf
from pyspark.sql.types import DoubleType

streaming_schema = StructType([
    StructField('Open', StringType(), True),
    StructField('High', StringType(), True),
    StructField('Low', StringType(), True),
    StructField('Close', StringType(), True),
    StructField('Adj Close', StringType(), True),
```

Step 14: After the Kafka cluster is set up on the instance, we will go to our Databricks notebooks and set up the cluster.
Go to compute and click ‘Create Compute’ to set up a cluster.



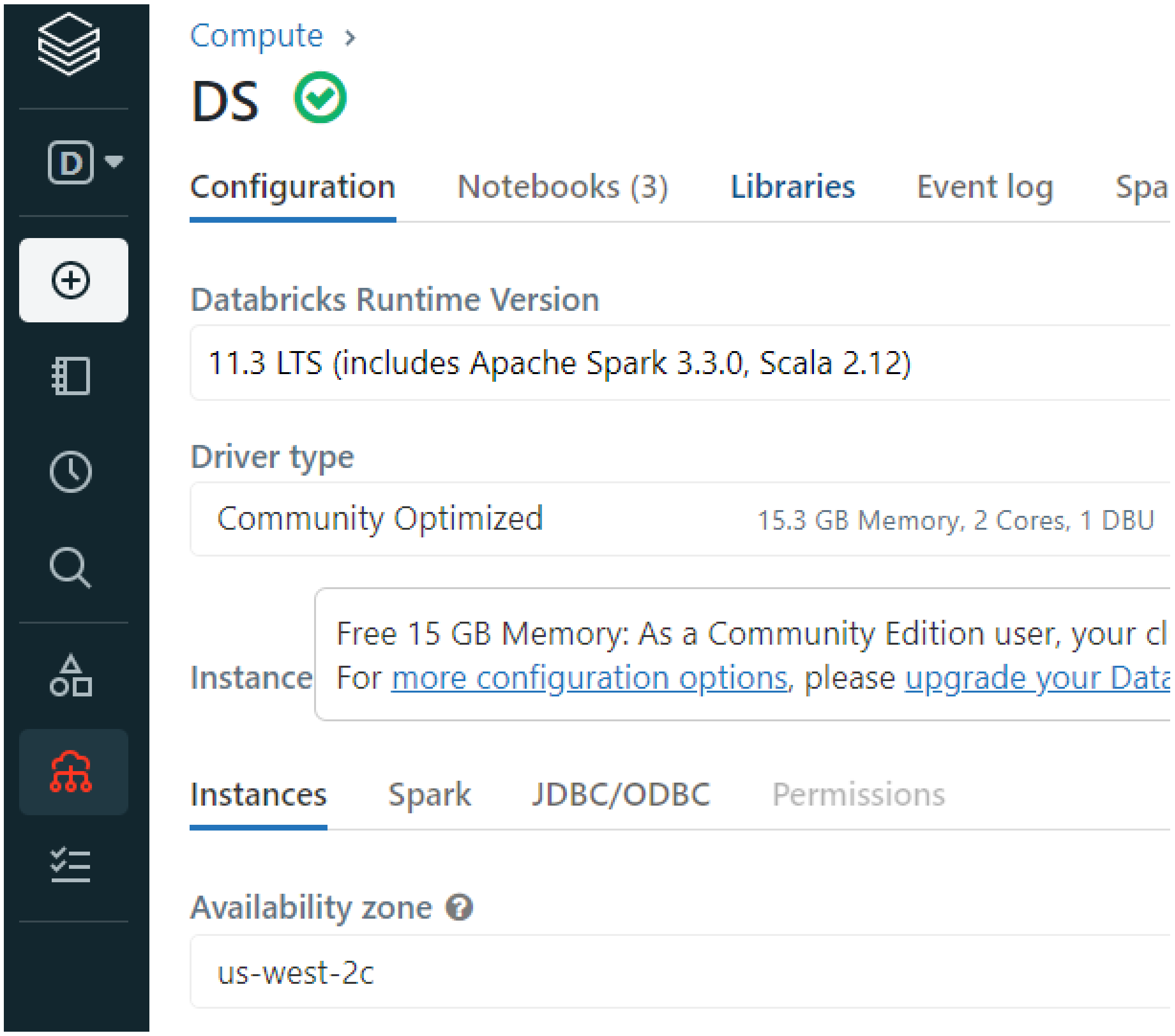
Compute

All-purpose compute Job compute

Create compute

	Name	Runtime	Active memory	Active cores	Active DBU / h	Source	Creator
✓	DS	11.3	15 GB	2 cores	1	UI	devarshsh
●	DS	11.3	-	-	-	UI	devarshsh
●	DS	11.3	-	-	-	UI	devarshsh

Step 15: Click on the cluster name and go to libraries



Databricks Runtime Version

11.3 LTS (includes Apache Spark 3.3.0, Scala 2.12)

Driver type

Community Optimized 15.3 GB Memory, 2 Cores, 1 DBU

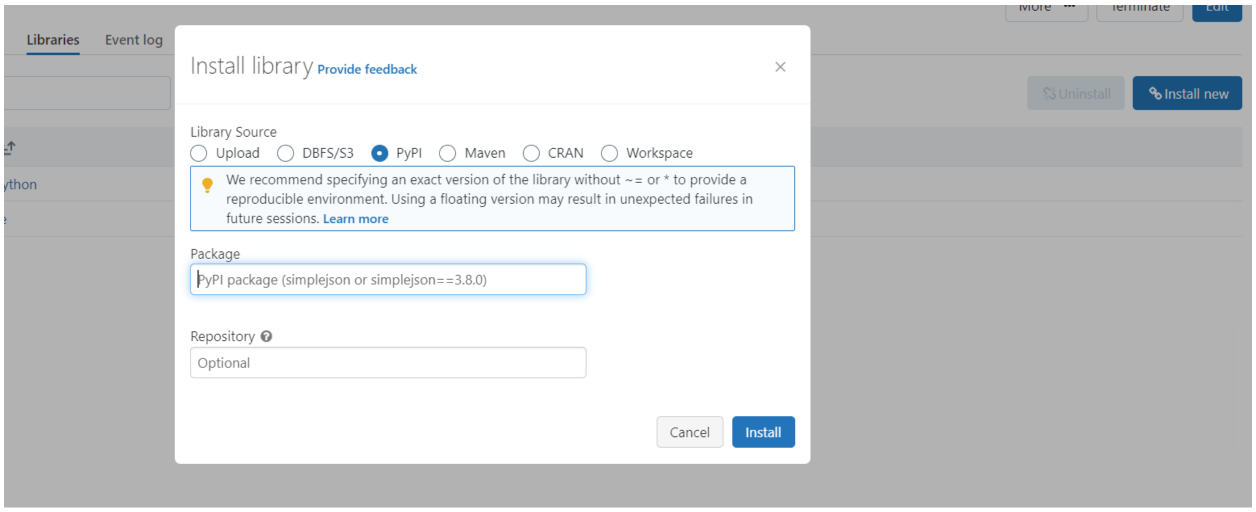
Instance

Free 15 GB Memory: As a Community Edition user, your cl
For more configuration options, please upgrade your Data

Availability zone

us-west-2c

Step 16: Select PyPI and install two libraries one by one, kafka-python and yfinance (Yahoo Finance)



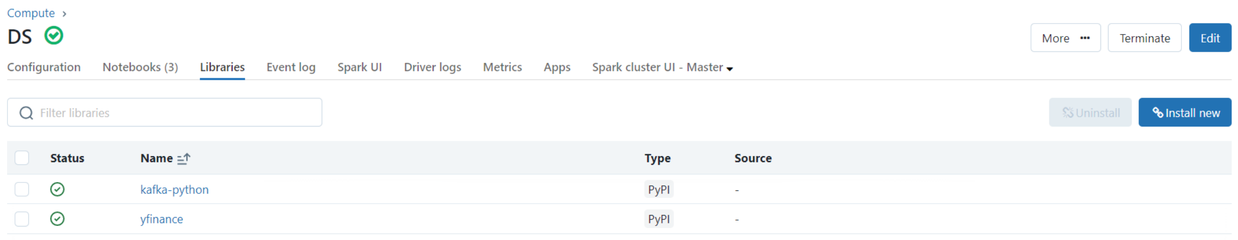
Install library Provide feedback

Library Source
Upload ( ) DBFS/S3 ( ) [✓ PyPI] Maven ( ) CRAN ( ) Workspace ( )

We recommend specifying an exact version of the library without ~= or * to provide a reproducible environment. Using a floating version may result in unexpected failures in future sessions. Learn more

Package: PyPI package (simplejson or simplejson==3.8.0)

Repository: Optional



Compute ›
DS

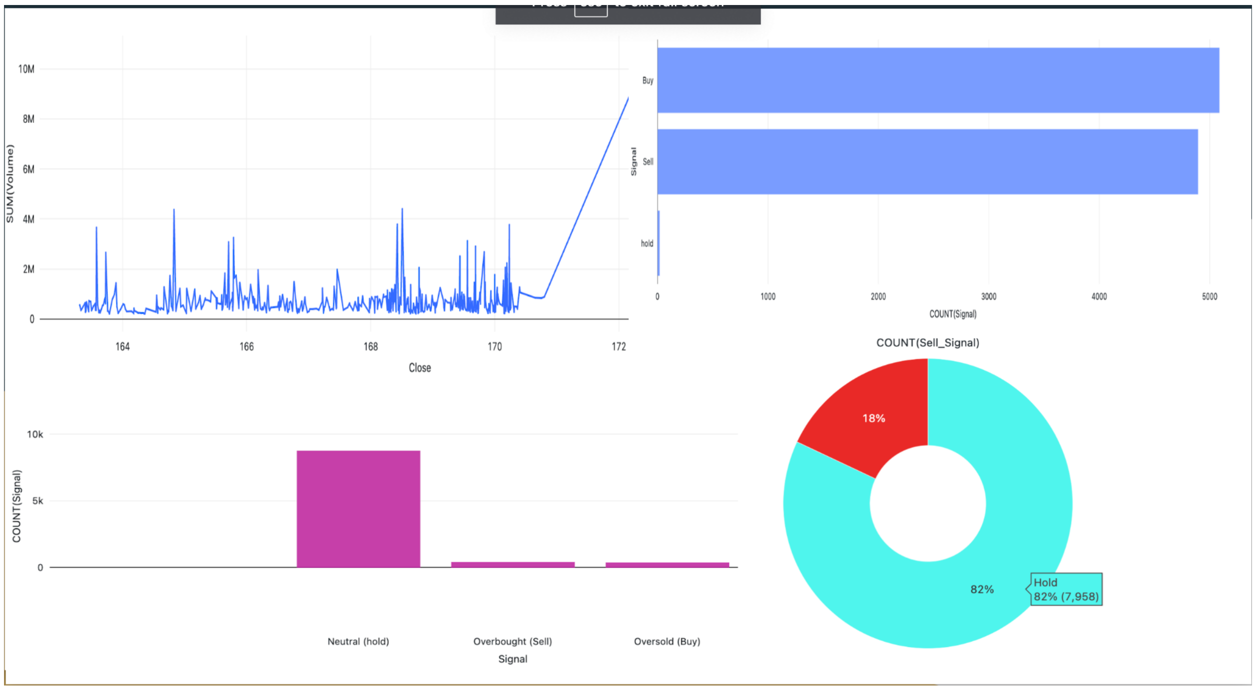
Status	Name	Type	Source
✓	kafka-python	PyPI	-
✓	yfinance	PyPI	-

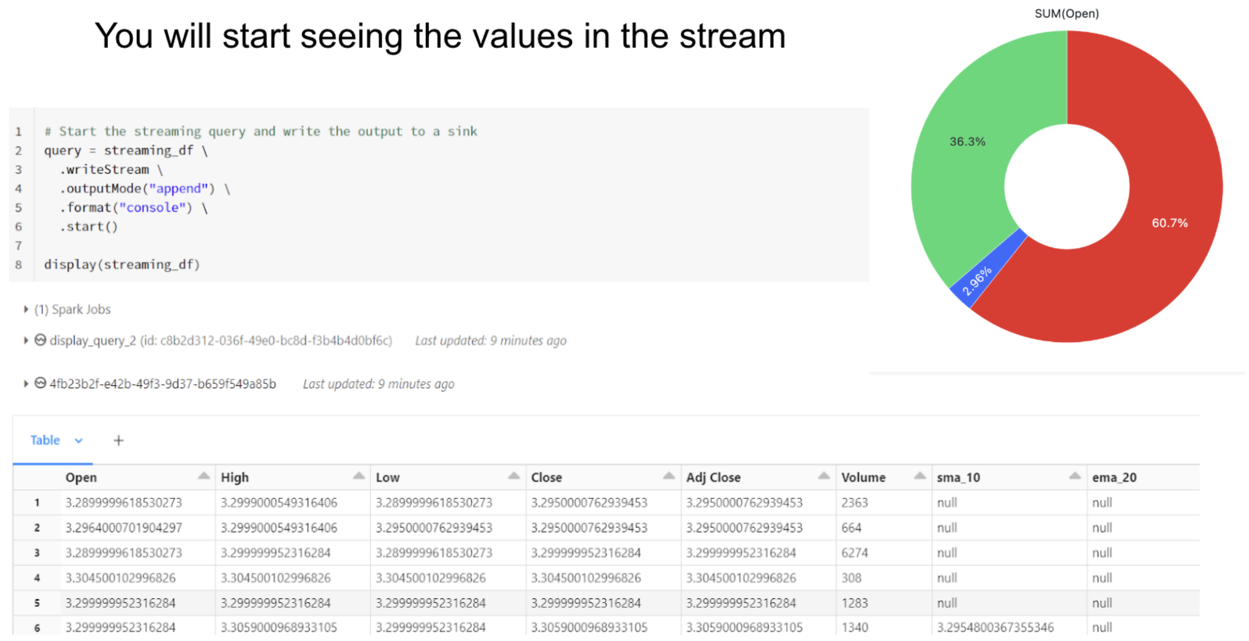
Steps to Run the Application

1. Open the AWS account and start the Kafka and Zookeeper by entering the correct credentials.
2. Open Databricks and create a cluster as mentioned above
3. Then install yfinance and Kafka-python libraries in that cluster
4. After that, create two files in the Databricks cluster, one for the consumer and another for the producer, and write the code (the whole code is provided in the Github repo)
5. Instead of creating two new files, we can simply upload the two ipynb files provided in GitHub and click on run all commands in the producer file after all the cells are executed successfully, then run all the commands in the consumer.
6. After running all the commands successfully, we can see the data is fetching from yahoo finance, and live streaming is happening.
7. In the consumer file, the data is streaming based on the logic we wrote in the producer file. The terms used in the logic are ema, sma, macd, and rsi. All these terms are used for doing technical analysis, which is explained in the section 3.
8. Secondly, you can also upload the CSV file, which you already have the data of stocks, directly into Databricks. The data file can be downloaded from the kagle.com. Here is the link: https://www.kaggle.com/datasets/hk7797/stock-market-india
9. For both ways, you can run the streaming of the data from yahoo finance and by uploading a CSV file.
10. Click on view visualization beside the streaming data table. You can see the graphical representation of the data. The screenshot is attached in the test results section.

Test Results

Below is the dashboard attached showing the buy and sell signal generated after applying the logic.




You will start seeing the values in the stream

```
# Start the streaming query and write the output to a sink
query = streaming_df \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .start()

display(streaming_df)
```

(1) Spark Jobs

display_query_2 (id: c8b2d312-036f-49e0-bc8d-f3b4b4d0bf6c) *Last updated: 9 minutes ago*

4fb23b2f-e42b-49f3-9d37-b659f549a85b *Last updated: 9 minutes ago*

	Open	High	Low	Close	Adj Close	Volume	sma_10	ema_20
1	3.2899999618530273	3.2999000549316406	3.2899999618530273	3.2950000762939453	3.2950000762939453	2363	null	null
2	3.2964000701904297	3.2999000549316406	3.2950000762939453	3.2950000762939453	3.2950000762939453	664	null	null
3	3.2899999618530273	3.299999952316284	3.2899999618530273	3.299999952316284	3.299999952316284	6274	null	null
4	3.304500102996826	3.304500102996826	3.304500102996826	3.304500102996826	3.304500102996826	308	null	null
5	3.299999952316284	3.299999952316284	3.299999952316284	3.299999952316284	3.299999952316284	1283	null	null
6	3.299999952316284	3.3059000968933105	3.299999952316284	3.3059000968933105	3.3059000968933105	1340	3.2954800367355346	null

References

- https://docs.aws.amazon.com/ec2/index.html
- https://docs.databricks.com/structured-streaming/index.html
- https://www.databricks.com/blog/2017/04/04/real-time-end-to-end-integration-with-apache-kafka-in-apache-sparks-structured-streaming.html
- https://kafka.apache.org/07/documentation.html
- https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/ema
- https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/rsi
- https://www.fidelity.com/bin-public/060_www_fidelity_com/documents/learning-center/tech-indicator-2023-slides.pdf
- https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/sma