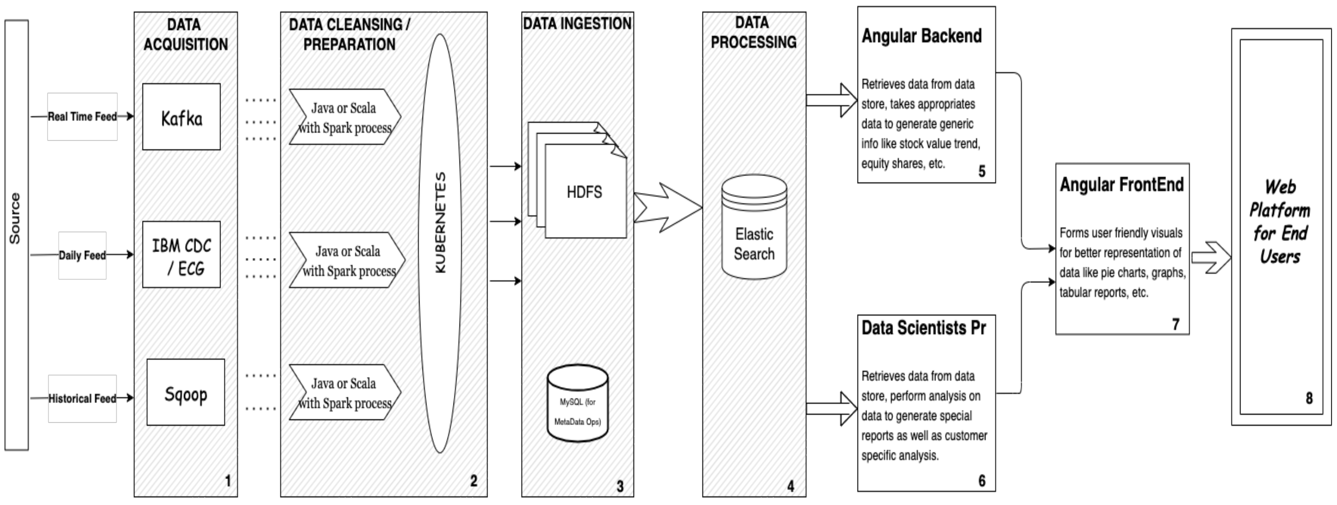
Stock Market Analytics Platform Architecture

This product is web based app that provides analytical solution of stock market. It gathers data from various stock market data providers through various acquisition channel depending on frequency of data.

Please find below the architecture diagram of the product:


Source
Real Time Feed
Daily Feed
Historical Feed
DATA ACQUISITION
Kafka
IBM CDC / ECG
Sqoop
1
DATA CLEANSING / PREPARATION
Java or Scala with Spark process
Java or Scala with Spark process
Java or Scala with Spark process
KUBERNETES
2
DATA INGESTION
HDFS
MySQL (for MetaData Ops)
3
DATA PROCESSING
Elastic Search
4
Angular Backend
Retrieves data from data store, takes appropriates data to generate generic info like stock value trend, equity shares, etc.
5
Data Scientists Pr
Retrieves data from data store, perform analysis on data to generate special reports as well as customer specific analysis.
6
Angular FrontEnd
Forms user friendly visuals for better representation of data like pie charts, graphs, tabular reports, etc.
7
Web Platform for End Users
8

DATA PIPELINE

<u>Set-up:</u>

Initially Source (Stock data provider) has to register for the application. It can be done via filling up a form in the web app with details like frequency of data, source connection details, name, etc. This gets stored into table "Active_Entities" in our relational database (MySQL) for quick access. It marks the validation point to start the above data ingestion process for active entities/sources.

Pipeline:

1. Based on the frequency of the data, we start the data acquisition process.

   a. For real time data, we use Kafka consumer as Kafka provides easy, fault tolerant streaming facilities with low latency and high throughput.

   b. For Incremental data, we can use either source API if they provide or third party tools like IBM CDC. In IBM CDC, we can configure the source destination and as soon as the configured amount of data is published to that destination or a specific time period is past, we capture the data to our process.

   c. For historical data, we can use Sqoop as it allows to transfer data from a variety of databases and is benefitted from parallelism of Hadoop.

2. This process will do some pre-scanning and standardization of data to be able to store & process further. Each consumer process is deployed and running on Kubernetes. Languages could Java or Scala with Spark so that we can utilize the inbuilt optimization and libraries of spark for distributed systems.

3. Since we are consuming data from a variety of sources, data will be in raw format and in huge volume. So we will be storing it in HDFS as it is known for its ability to be able to process the data in parallel manner and is well suited for BigData. The meta data would be stored in relational database MySQL for quick access.

4. We will now filter out the relevant data and put it in ElasticSearch which is document-oriented datastore and has following benefits:

    - One can perform and combine various kind of searches irrespective of their data type which included structured, unstructured, geo and metrics data type.
    - Query can be retrieved data in any form required.
    - Possible to analyze billions of records in few seconds.
    - It also provides aggregations which can explore trends and patterns of data.
    - And it is suited for full-text search feature also.

5. The web backend can retrieve the data using ElasticSearch API and form required aggregations like daily trends, price increase/decrease, etc.

6. The Data Science Team also can retrieve required data using ElasticSearch API and generate special reports which can be published to UI.

7. The frontend part can form meaningful charts and user friendly graphics to be displayed on the web page.

8. Finally, we can browse the information & analysis on the web app and request for any specific analysis.

<u>Other details:</u>

- The UI will have an option where user can request any specific analysis which will go as a mail to the Data Science team and will be published to the UI after completion of the report.
- Integration of ServiceNow portal for incident tracking and ensuring SLAs are met. For example, if user wants to report any bug, they can raise a ticket to the dev team. The ServiceNow Portal will ensure that mail is sent to the required team and reminders are also being sent to ensure that SLAs are met.
- HDFS replicates the data and has the default replication factor to ensure fault tolerance behaviour and hence keeping disaster recovery checked.