Task → Install Kafka on AWS EC2 ✓

ML System

Kafka

- Distributed
- fault-tolerant
- high-throughput

Streaming platform

→ Real-time data feeds

→ Not a simple message-Queue

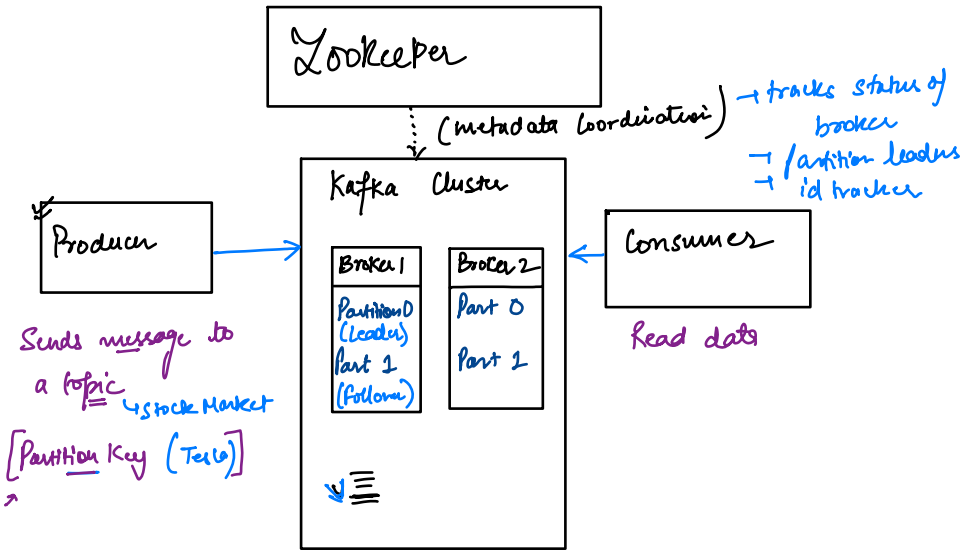
- distributed
- Optimised for horizontal Scaling
- durable
- low-latency ✓

(1) Persistent Storage

(2) Scalable → Add brokers

(3) Fault tolerant →

(4) Ordering gaurantee



Machine learning System design for Real time Stock market Analysis

Functional Req

1. Real-time data ingestion
   ↳ ingest prices, volume, metadata every 1-sec
2. Stream processing → clean, aggregate & transform data in real time.
3. Batch & real-time Storage →
   Store raw & processed data in query-optimized format.

④ Real-time Alerts → Detect anomaly (volume spikes, price drops)

⑤ Batch Analytics → SQL queries on historical data

⑥ ML predictions → Predict short-term stock trends

⑦ Dashboards → Visualize trends

Non funct^{n}

① low-latency → < 5 sec [End to end] data ingest^{n} → prediction

② Scalability → 1000+ stocks with 1 sec updates

③ fault tolerance

④ Cost efficiency. [S3, EC2] VM

① Stock Data Source (yahoo finance, Alpha Vantage API)

↓

② Kafka [EC2] ------ Real-time Alerting [Lambda]

↓

③ Raw Data Lake [S3]

Load Kafka on EC2

1. Launch an EC2 instance → Amazon Linux 2
2. SSH client details
3. Make a connection using Key-value pair
   - local terminal
   - SSH command

1) `sudo yum install java`
2) `wget https://archive.apache.org/dist/kafka/3.3.1/kafka_2.12-3.3.1.tgz` (download Kafka)
3) `tar -xzf kafka_2.12-3.3.1.tgz` (unarchive)
4) `cd kafka_2.12-3.3.1/` (go inside the folder)
5) Kafka uses Zookeeper for distributed coordination. Start Zookeeper in the background:

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

6) Open new terminal

   Increase memory → `export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"`

7) Start the Kafka server:

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

3

Raw S3 file (csv / parquet / json)

lhaha

[s3:// stock-data / raw / date=2023-10-10 / hour=12 / lhaha.csv]

partitions = date hour

AWS Glue ETL

4

Example

Job1 Convert JSON to parquet, sanitize data (handle missing value)

Job2 feature engineering (moving Avg, RSI, MACD)

⇓

5 [S3://stock-data/processed/date/hour/—

6 SQL for analytics

AWS Athena

```
-- Create a table referencing S3 data
CREATE EXTERNAL TABLE my_table (
id INT,
name STRING,
age INT
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 's3://my-bucket/data/';
```

7 optional → Alerts → AWS Lambda

```
def lambda_handler(event, context):
    for record in event['records']:
        data = json.loads(record['value'])
        if data['volume'] > 1000000:
            sns.publish(TopicArn='arn:aws:sns:...', Message=f"Volume spike: {data['symbol']}")
```

SNS alert → mailbox

(8) Feature engineering

+ Moving Avg

```
df['5min_MA'] = df['close'].rolling(window=300).mean()  # 300 seconds = 5 mins
```

(9) Modelling

Model	Pros	Cons	Best Used
XGBOOST	Fast, Tabular data ✓	Manual feature Engineering	Intraday predictions
Prophet	Handles seasonality, easy to use	Struggle Volatile stocks	Long term trends
LSTM	Handle temporal patterns	Needs large data, slow train	High-frequency trading ✓

(10) Minute level predictions

Deploy → Sagemaker endpoint

→ Predictions to Kafka topic

↳ predictions for real-time dashboards.

```
model = Sequential([
    LSTM(128, input_shape=(60, 5),  # 60 time steps, 5 features
    Dropout(0.3),
    Dense(1)
])
model.compile(loss='mae', optimizer='adam')
```

Problem → Prediction w/ sec

LSTM can take time

→ [Prophet → for alerts]