

بسم الله الرحمن الرحيم



سیستم تحلیل مالی و معاملات بلادرنگ با هوش مصنوعی

پروژه نهایی درس تحلیل‌ها و سیستم‌های داده حجیم



استاد درس: دکتر حسن نادری

دانشجویان: علی احمدی، امیر کریمی، مهدی شکاری و اهورا امینی

بهمن ۱۴۰۳

فهرست مطالب

2.....	مقدمه
2.....	هدف پروژه
2.....	اهمیت پروژه
2.....	چالش‌ها
3.....	معماری سیستم
4.....	API Nobitex
4.....	Data Ingestion Service
4.....	Apache Kafka
4.....	Stream Data Processing Service (Apache Spark)
5.....	محاسبه شاخص‌ها
5.....	تولید سیگنال‌ها
6.....	پردازش میکروبیج‌ها
6.....	خواندن و نوشتن با Spark
6.....	مدیریت استریم
6.....	سرویس میانی Kafka-to-QuestDB
6.....	پردازش پیام‌ها
6.....	جایگزینی nan و تبدیل مقادیر
7.....	ساخت قالب ILP و ارسال داده‌ها به QuestDB
7.....	QuestDB
8.....	Aggregation Service
8.....	Aggregate API
9.....	Summarize API
9.....	Summarize Multiple API
10	Visualization Service (Grafana)

مقدمه

هدف پروژه

پروژه تحلیل داده‌های مالی در بازارهای مالی مانند رمزارز و نمادهای سهام مختلف به منظور تجزیه و تحلیل داده‌ها در زمان واقعی (Real-time) طراحی و پیاده‌سازی شده است. این سیستم با استفاده از تکنولوژی‌های توزیع شده و پردازش بلادرنگ داده‌ها، امکان تحلیل سریع و تصمیم‌گیری به موقع برای معامله‌گران بازارهای مالی را فراهم می‌کند.

بازارهای مالی به دلیل نوسانات شدید و لحظه‌ای قیمت‌ها، نیاز به ابزارهایی دارند که بتوانند به سرعت و به صورت دقیق، اطلاعات را پردازش کرده و سیگنال‌های مناسب برای خرید و فروش ارائه دهند. در این پروژه، هدف اصلی این است که با استفاده از پردازش‌های پیچیده و بلادرنگ، سیگنال‌های دقیق و به موقع برای معامله‌گران فراهم شود تا بتوانند تصمیمات بهتری اتخاذ نمایند.

اهمیت پروژه

تحلیل داده‌های مالی از اهمیت بسیار بالایی برخوردار است. با افزایش حجم داده‌ها و نیاز به پردازش بلادرنگ، استفاده از سیستم‌های مقیاس‌پذیر و توزیع شده مانند Apache Spark، Apache Kafka و QuestDB به ویژه در تحلیل‌های مالی اهمیت پیدا می‌کند. این پروژه نه تنها بر مقیاس‌پذیری تمرکز دارد، بلکه توانایی پردازش داده‌ها در لحظه را نیز فراهم می‌آورد.

به طور ساده، این پروژه یک سیستم بلادرنگ است که به صورت خودکار، داده‌ها را از بازارهای مالی دریافت کرده، پردازش کرده و سپس سیگنال‌های مناسب خرید و فروش و همچنین با استفاده از هوش مصنوعی پیشبینی روند را نیز به کاربران نمایش می‌دهد.

این توانایی می‌تواند برای کسانی که در بازارهای مالی فعالیت می‌کنند، به ویژه برای کسانی که به دنبال تصمیم‌گیری‌های سریع و بهینه هستند، بسیار ارزشمند باشد.

چالش‌ها

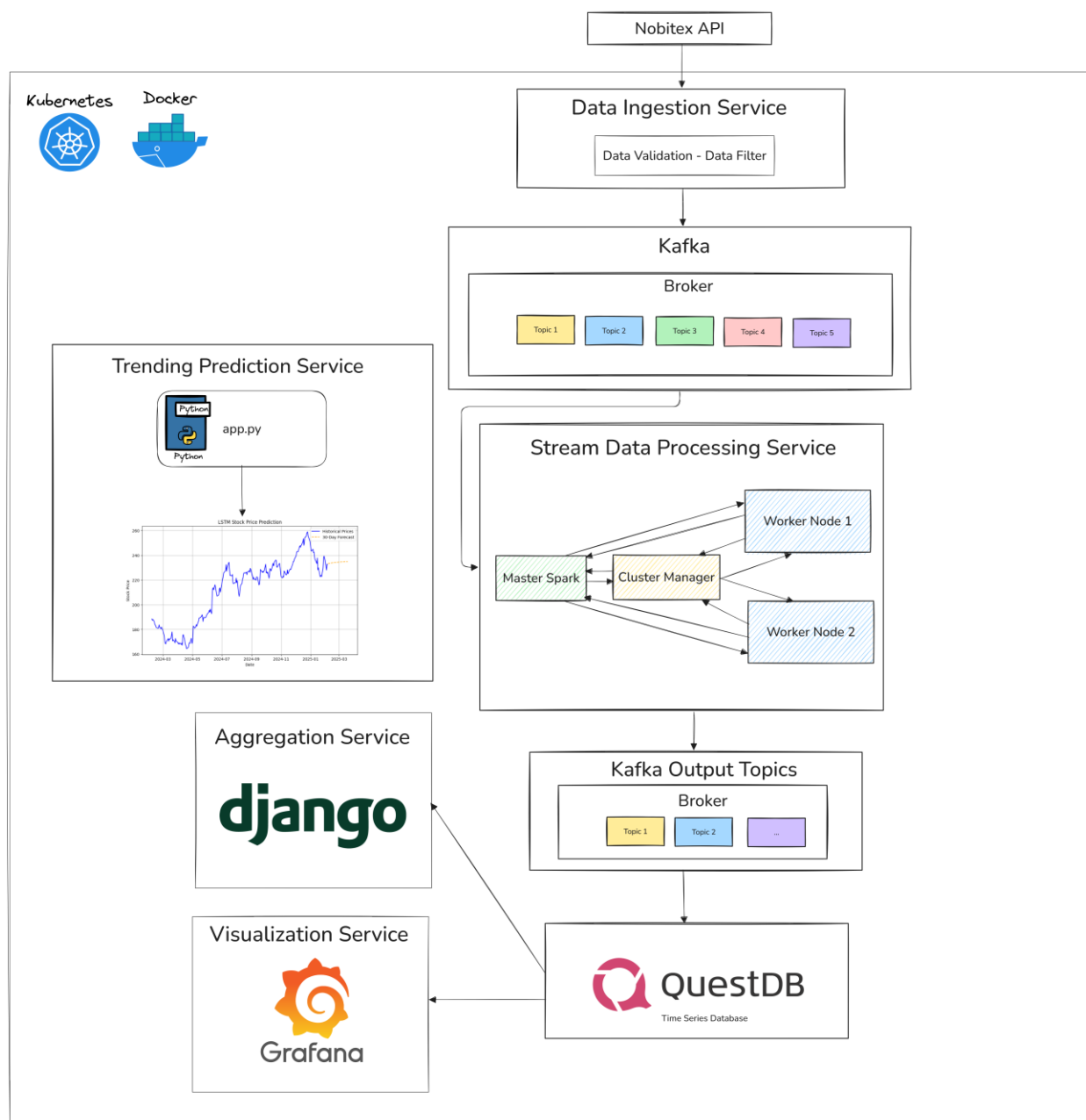
در این پروژه، برخی از چالش‌ها به شرح زیر هستند:

- **حجم بالای داده‌ها:** با توجه به حجم عظیم داده‌های موجود در بازارهای مالی، پردازش سریع داده‌ها به صورت بلادرنگ نیازمند استفاده از سیستم‌های توزیع شده است.
- **تاخیر در پردازش (Latency):** سیستم باید قادر باشد تا داده‌ها را با کمترین تاخیر پردازش کرده و سیگنال‌های دقیق و به موقع برای کاربران ارسال نماید.
- **مقیاس‌پذیری:** با توجه به اینکه داده‌ها به صورت لحظه‌ای وارد سیستم می‌شوند، مقیاس‌پذیری سیستم برای مدیریت تعداد زیاد درخواست‌ها و داده‌ها ضروری است.
- **پیشبینی روند با استفاده از هوش مصنوعی:** به دلیل پیچیدگی ذاتی داده‌های مالی و همچنین پیشبینی آن‌ها، استفاده از الگوریتم مناسب برای تحلیل روند در آینده بسیار چالش برانگیز و محل بحث است.

هدف از این پروژه این است که با استفاده از معماری‌های مقیاس‌پذیر، پردازش‌های توزیع شده و استفاده از هوش مصنوعی (یادگیری عمیق) این چالش‌ها را پشت سر گذاشته و سیستمی کارآمد و سریع برای تحلیل داده‌های بازارهای مالی ارائه دهد.

معماری سیستم

معماری سیستم به شرح زیر است:



این معماری یک سیستم توزیع شده برای پردازش داده‌ها به صورت بلادرنگ است که شامل اجزای مختلفی می‌باشد. در ادامه توضیحات معماری ارائه شده به تفصیل شرح داده خواهد شد.

API Nobitex

این API به عنوان منبع داده‌های رمز ارز عمل می‌کند. داده‌ها شامل اطلاعات معاملات، قیمت‌ها، حجم بازار و سایر معیارهای مالی مرتبط با رمزارزها می‌باشد. ساختار داده‌های دریافتی به شرح زیر است:

نام متغیر	نوع متغیر	توضیحات
stock_symbol	object	نماد سهام
local_time	timestamps	تاریخ و ساعت دقیق دریافت داده
open_prices	float	قیمت باز شدن (بر اساس کندل)
high_prices	float	بالاترین قیمت
low_prices	float	پایین‌ترین قیمت
close_prices	float	قیمت بسته شدن
volumes	int	حجم معامله بازار

Data Ingestion Service

وظیفه این سرویس جمع‌آوری و انتقال داده از API Nobitex است. در این فرآیند، داده‌ها مورد اعتبارسنجی (Validation) و فیلترگذاری قرار می‌گیرند تا فقط داده‌های معتبر به مرحله بعدی ارسال شوند. در صورت صحت داده‌ها، توسط کد producer هر داده جدید بر اساس فیلد stock_symbol به یک Topic در Kafka ارسال می‌شود.

برای مثال داده‌های مربوط به BTCIRT به تاپیک btcirt_topic ارسال خواهد شد.

```
2025-01-25 18:35:44 INFO:root:Data sent to topic=shibirt_topic, partition=0, offset=7, data={'stock_symbol': 'SHIBIRT', 'local_time': '2025-01-25 18:34:00', 'open': 1680.2, 'high': 1684.6, 'low': 1680.1, 'close': 1680.1, 'volume': 811.0, 'topic': 'shibirt_topic'}
2025-01-25 18:35:44 INFO:root:payload: {'stock_symbol': 'SHIBIRT', 'local_time': '2025-01-25 18:34:00', 'open': 1680.2, 'high': 1684.6, 'low': 1680.1, 'close': 1680.1, 'volume': 811.0, 'topic': 'shibirt_topic'} processed and forwarded to Kafka topic: shibirt_topic
2025-01-25 18:36:45 INFO:root:Data sent to topic=btctrt_topic, partition=0, offset=7, data={'stock_symbol': 'BTCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 8697888889.0, 'high': 8697888889.0, 'low': 8697888889.0, 'close': 8697888889.0, 'volume': 5.22939e-05, 'topic': 'btctrt_topic'}
2025-01-25 18:36:45 INFO:root:payload: {'stock_symbol': 'BTCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 8697888889.0, 'high': 8697888889.0, 'low': 8697888889.0, 'close': 8697888889.0, 'volume': 5.22939e-05, 'topic': 'btctrt_topic'} processed and forwarded to Kafka topic: btctrt_topic
2025-01-25 18:36:53 INFO:root:Data sent to topic=usdtirt_topic, partition=0, offset=8, data={'stock_symbol': 'USDIRT', 'local_time': '2025-01-25 18:36:00', 'open': 83597.0, 'high': 83598.0, 'low': 83552.0, 'close': 83553.0, 'volume': 2718.0373713987, 'topic': 'usdtirt_topic'}
2025-01-25 18:36:53 INFO:root:payload: {'stock_symbol': 'USDIRT', 'local_time': '2025-01-25 18:36:00', 'open': 83597.0, 'high': 83598.0, 'low': 83552.0, 'close': 83553.0, 'volume': 2718.0373713987, 'topic': 'usdtirt_topic'} processed and forwarded to Kafka topic: usdtirt_topic
2025-01-25 18:36:54 INFO:root:Data sent to topic=ethirt_topic, partition=0, offset=8, data={'stock_symbol': 'ETHIRT', 'local_time': '2025-01-25 18:36:00', 'open': 277734600.0, 'high': 277734600.0, 'low': 277170000.0, 'close': 277734600.0, 'volume': 0.2178, 'topic': 'ethirt_topic'}
2025-01-25 18:36:54 INFO:root:payload: {'stock_symbol': 'ETHIRT', 'local_time': '2025-01-25 18:36:00', 'open': 277734600.0, 'high': 277734600.0, 'low': 277170000.0, 'close': 277734600.0, 'volume': 0.2178, 'topic': 'ethirt_topic'} processed and forwarded to Kafka topic: ethirt_topic
2025-01-25 18:36:57 INFO:root:Data sent to topic=etctrt_topic, partition=0, offset=8, data={'stock_symbol': 'ETCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 2261992.0, 'high': 2261992.0, 'low': 2261992.0, 'close': 2261992.0, 'volume': 0.0, 'topic': 'etctrt_topic'}
2025-01-25 18:36:57 INFO:root:payload: {'stock_symbol': 'ETCIRT', 'local_time': '2025-01-25 18:36:00', 'open': 2261992.0, 'high': 2261992.0, 'low': 2261992.0, 'close': 2261992.0, 'volume': 0.0, 'topic': 'etctrt_topic'} processed and forwarded to Kafka topic: etctrt_topic
```

Apache Kafka

داده‌های تصحیح شده توسط سرویس Data Ingestion به Kafka به عنوان یک سیستم مقیاس‌پذیر و پایدار وارد می‌شوند. امکان انتقال داده‌ها به اجزای مختلف را به شیوه‌ای بسیار کارآمد فراهم می‌کند.

Broker: مسئول توزیع داده‌ها در قالب Topic‌های مختلف است. در معماری سیستم چندین Topic نمایش داده شده است که هر کدام ممکن است یک جزئی از داده را نمایش دهند.

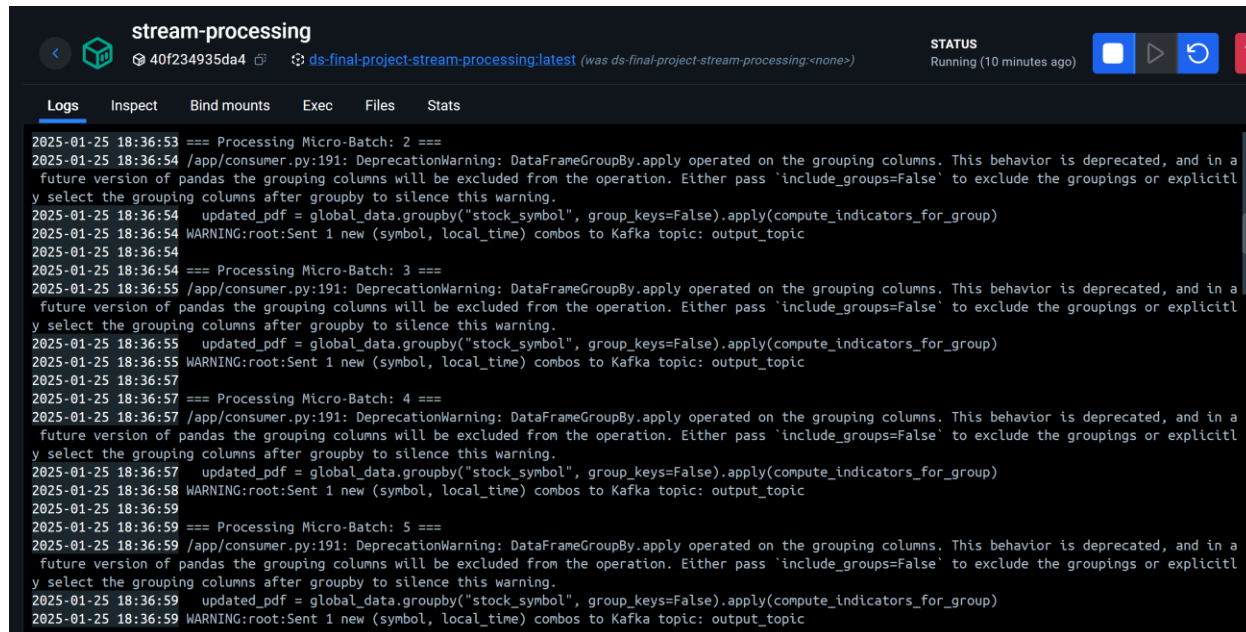
Stream Data Processing Service (Apache Spark)

داده‌های ارسال شده از Kafka در اینجا پردازش می‌شوند. Apache Spark به عنوان یک محیط پردازش موازی برای انجام عملیات پیچیده روی داده‌ها عمل می‌کند.

Cluster Spark و Master Spark: مدیریت عملیات پردازش توزیع شده را برعهده دارند که داده‌ها میان Worker Node1 و Worker Node2 تقسیم می‌شوند.

Kafka Output Topics: داده‌های پردازش شده مجدداً به Kafka ارسال می‌شوند، جایی که موضوعات خروجی جدید برای این داده‌ها ایجاد می‌شود.

این سرویس شامل چندین بخش است که به طور کلی برای پردازش و داده‌های استریم از Kafka، محاسبه شاخص‌های معاملاتی یا همان اندیکاتورها (SMA, EMA, RSI) و ارسال داده به Kafka و مدیریت میکروبیج‌ها با Spark طراحی شده است.



```
stream-processing
40f234935da4 ds-final-project-stream-processing:latest (was ds-final-project-stream-processing:<none>)
STATUS
Running (10 minutes ago)

Logs Inspect Bind mounts Exec Files Stats

2025-01-25 18:36:53 === Processing Micro-Batch: 2 ===
2025-01-25 18:36:54 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:54 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:54 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
2025-01-25 18:36:54 === Processing Micro-Batch: 3 ===
2025-01-25 18:36:55 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:55 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:55 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
2025-01-25 18:36:57 === Processing Micro-Batch: 4 ===
2025-01-25 18:36:57 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:57 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:58 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
2025-01-25 18:36:59 === Processing Micro-Batch: 5 ===
2025-01-25 18:36:59 /app/consumer.py:191: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass 'include_groups=False' to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
2025-01-25 18:36:59 updated_pdf = global_data.groupby("stock_symbol", group_keys=False).apply(compute_indicators_for_group)
2025-01-25 18:36:59 WARNING:root:Sent 1 new (symbol, local_time) combos to Kafka topic: output_topic
```

محاسبه شاخص‌ها

این بخش برای هر گروه از داده‌ها شاخص‌های زیر را محاسبه می‌کند:

- **SMA (5)**: میانگین ساده در دوره‌های ۵ دقیقه‌ای
- **EMA (10)**: میانگین نمایی دوره‌های ۱۰ دقیقه‌ای
- **RSI (10)**: شاخص قدرت نسبی در دوره‌های ۱۰ دقیقه‌ای
- **Average gain & Average Loss**: در دوره‌های ۱۰ دقیقه‌ای

تولید سیگنال‌ها

در این بخش سیگنال‌های خرید، فروش یا نگه داشتن بر اساس شرایط زیر تولید می‌شود:

- **سیگنال خرید**: اگر $EMA(10) > SMA(5)$ باشد و $RSI(10)$ نیز کمتر از ۷۰ باشد.
- **سیگنال فروش**: اگر $EMA(10) < SMA(5)$ باشد و $RSI(10)$ بیشتر از ۳۰ باشد.
- در غیر این صورت سیگنال «نگه داشتن» یا همان hold است.

پردازش میکروبیج‌ها

هر میکروبیج داده‌های زیر را پردازش می‌کند:

- تبدیل داده‌های Spark به Pandas
- محاسبه شاخص‌ها و سیگنال‌ها
- فیلتر کردن داده‌هایی که قبلاً ارسال شده‌اند.
- ارسال داده‌های جدید به Kafka

خواندن و نوشتن با Spark

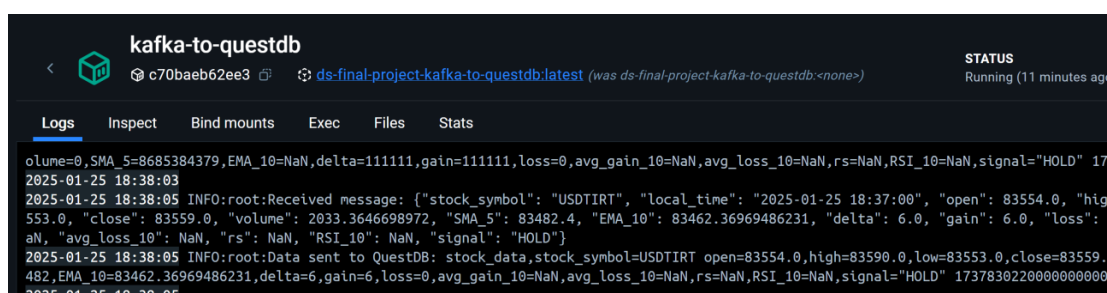
- داده‌ها از Kafka با فرمت Json خوانده می‌شوند و با استفاده از foreachBatch به صورت دسته‌ای پردازش می‌شوند.
- یک اسکیمای مشخص برای ستون‌های داده تعریف شده است.

مدیریت استریم

داده‌ها به صورت استریم از Kafka خوانده شده، شاخص‌ها محاسبه شده و داده‌های نهایی مجدداً به output_topic ارسال می‌شوند.

سرویس میانی Kafka-to-QuestDB

این سرویس برای دریافت داده‌ها از Kafka، پردازش آن‌ها و ارسال آن‌ها به QuestDB از طریق Influx Line Protocol یا ILP طراحی شده است. در ادامه بخش‌های مختلف آن توضیح داده می‌شود.



The screenshot shows the 'kafka-to-questdb' application interface. At the top, there's a status bar indicating it's 'Running (11 minutes ago)'. Below this, there are tabs for 'Logs', 'Inspect', 'Bind mounts', 'Exec', 'Files', and 'Stats'. The 'Logs' tab is selected, displaying a log entry from 2025-01-25 18:38:05. The log message is an INFO from the root, stating 'Received message: {"stock_symbol": "USD TIRT", "local_time": "2025-01-25 18:37:00", "open": 83554.0, "high": 83559.0, "close": 83559.0, "volume": 2033.3646698972, "SMA_5": 83482.4, "EMA_10": 83462.36969486231, "delta": 6.0, "gain": 6.0, "loss": -1.0, "avg_loss_10": NaN, "rs": NaN, "RSI_10": NaN, "signal": "HOLD"}'. Below this, another log entry shows 'Data sent to QuestDB: stock_data, stock_symbol=USD TIRT open=83554.0, high=83590.0, low=83553.0, close=83559.0, volume=2033.3646698972, SMA_5=83482.4, EMA_10=83462.36969486231, delta=6, gain=6, loss=0, avg_gain_10=NaN, avg_loss_10=NaN, rs=NaN, RSI_10=NaN, signal="HOLD" 1737830220000000000'.

- **Kafka Consumer**: برای دریافت پیام‌ها از Kafka
- **Socket**: برای ارتباط با QuestDB از طریق پروتکل ILP
- **کتابخانه‌های Pandas و math**: برای پردازش داده‌ها

پردازش پیام‌ها

- پیام‌ها از Kafka دریافت و مقدار آن‌ها (JSON) پردازش می‌شود.
- اگر پیام شامل داده‌های نامعتبر باشد، خطا گزارش می‌شود.

جایگزینی nan و تبدیل مقادیر

- مقدار "NaN" با مقدار float('nan') جایگزین می‌شود.
- مقادیر عددی مانند volume و سایر شاخص‌ها به نوع مناسب تبدیل می‌شوند.

ساخت قالب ILP و ارسال داده‌ها به QuestDB

- داده‌ها در قالب Influx Line Protocol برای QuestDB قالب‌بندی می‌شوند.
- هر رکورد شامل اطلاعاتی مثل قیمت‌ها (open, close,...) که بیشتر در ابتدا آمده بود و شاخص‌های محاسبه شده است.

QuestDB

داده‌های پردازش شده در QuestDB ذخیره می‌شوند. QuestDB یک پایگاه داده سری زمانی است که برای انجام تحلیل‌های پیچیده و بلادرنگ استفاده می‌شوند. این پایگاه داده برای داده‌های بازارهای مالی نیز بسیار مناسب است و stock_symbol را به صورت خاص index گذاری می‌کند که به صورت سریع بتواند داده‌های مربوط به هر نماد سهام را در کوئری‌ها برگرداند.

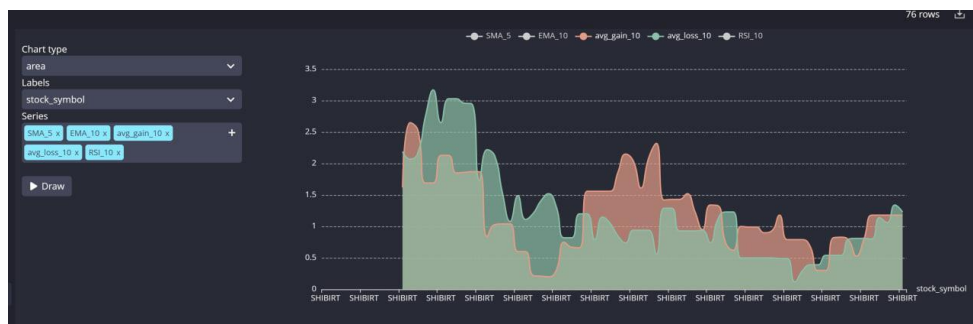
استفاده از این دیتابیس بسیار بهینه‌تر از ترکیب یک دیتابیس دیگر مثل PostgreSQL با Redis است زیرا:

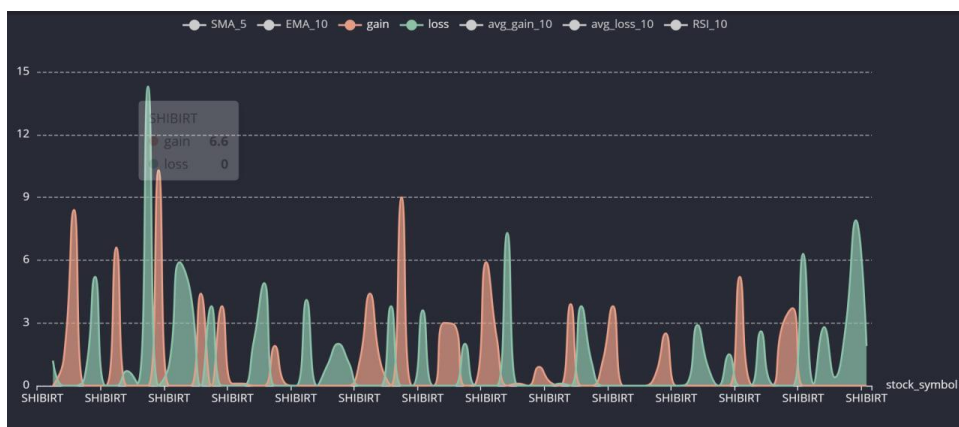
- **عملکرد بالا:** QuestDB برای پردازش داده‌های سری زمانی بهینه شده و می‌تواند حجم عظیمی از داده‌ها را با سرعت بسیار بالا وارد کوئری کند.
- **کاهش پیچیدگی معماری:** با استفاده از QuestDB دیگر نیازی به ترکیب چندین ابزار نیست.
- **مقیاس‌پذیری:** QuestDB در مدیریت حجم عظیمی از داده‌های سری زمانی مقیاس‌پذیری بالایی دارد، در حالی که Redis برای داده‌های بسیار بزرگ نیاز به حافظه بیشتر و تنظیمات پیچیده‌تری دارد.
- **پشتیبانی از ILP:** امکان ارسال داده‌ها با فرمت ILP باعث ساده‌تر شدن ارسال داده‌های بازار سهام می‌شود.
- جستجو و انجام کوئری‌های پیشرفته SQL: قابلیت استفاده از SQL استاندارد برای کوئری‌های پیچیده مانند محاسبه میانگین، محاسبه شاخص‌های معاملاتی (EMA, SMA, ...) و جستجوی مقادیر خاص در بازه‌های زمانی
- **پشتیبانی از داده‌های بلادرنگ (Realtime):** مناسب برای سیستم‌های بلادرنگ که داده‌های بازار سهام را در لحظه پردازش و تحلیل می‌کنند.
- **مقیاس‌پذیری بالا:** مناسب برای ذخیره و پردازش داده‌های حجیم مانند داده‌های تاریخی بازار سهام
- **پشتیبانی از چارت‌های مختلف برای تحلیل بصری داده**

در زیر تصویری از محیط دیتابیس نمایش داده شده است:

stock_symbol	signal	local_time	open	close	high	low	volume	SMA_5	EMA_10	delta
SHIBIRT	SELL	2025-01-25T17:37:00.000000Z	1680	1680	1680	1680	91	1677	1678.679698218301	-
BTCTRT	SELL	2025-01-25T17:38:00.000000Z	8711138043	8711138042	8711138043	8711138042	0	8711136842	8714378211.468225	-1
USDTRT	SELL	2025-01-25T17:38:00.000000Z	83766	83695	83788	83695	809	83736	83760.10819324121	-1
ETHTRT	BUY	2025-01-25T17:39:00.000000Z	278199995	278199995	278199995	278199995	0	278098468	278081127.14528024	-
ETCTRT	BUY	2025-01-25T17:39:00.000000Z	2255017	2255017	2255017	2255017	0	2257809	2257796.158471043	4
SHIBIRT	SELL	2025-01-25T17:39:00.000000Z	1673.9	1673.9	1673.9	1673.9	0	1677	1677.81066217861	-4
BTCTRT	HOLD	2025-01-25T17:40:00.000000Z	8711138043	8711138043	8711138043	8711138043	0	8711137442	8713782544.474003	-
USDTRT	SELL	2025-01-25T17:40:00.000000Z	83712	83792	83792	83712	322	83736	83765.90686719735	9
ETHTRT	BUY	2025-01-25T17:40:00.000000Z	278199000	278199000	278199000	278199000	0	278198127	278069831.46425474	-99
ETCTRT	BUY	2025-01-25T17:40:00.000000Z	2252010	2252010	2252010	2252010	0	2257207	2256744.1296581263	-300
SHIBIRT	SELL	2025-01-25T17:40:00.000000Z	1675.1	1675.1	1675.1	1675.1	59	1676	1677.317814509771	-

در زیر تصویری از چارت‌های QuestDB نمایش داده شده است:





Aggregation Service

در این سرویس از یک اپلیکیشن جنگو استفاده شده است که شامل ۳ نوع API است.

Aggregate API

در این API با نوشتن اسم سهام مورد نظر، نوع aggregation شامل میانگین‌گیری، بیشترین یا کمترین مقدار، بازه زمانی مورد نظر و فیلد انتخابی مورد نظر جواب مناسب خواهید گرفت. برای مثال:

```
POST http://localhost:8000/api/aggregate/

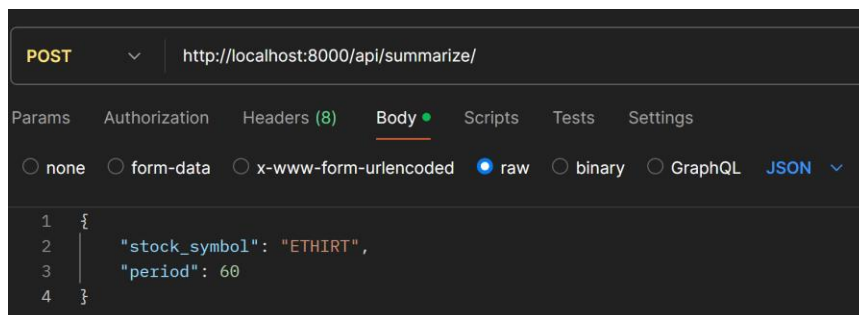
{
  "aggregation": "highest", // "highest", "lowest", "avg"
  "stock_symbol": "ETHIRT",
  "period": 10, // (in minutes). e.g. from 10 minutes ago until now
  "field": "close" // any field in the database with type 'double'
}
```

```
{
  "highest": 277734600.0,
  "field": "close",
  "stock_symbol": "ETHIRT"
}
```

Summarize API

خلاصه عملکرد یک سهام در بازه زمانی مشخص را برای معیارهای close, gain, loss, RSI (10), EMA (10), SMA (5) تولید می‌کند.

برای مثال ورودی:



و خروجی آن:

```
4  "summary": {
5    "close": {
6      "avg": 277430236.3636364,
7      "highest": 277800000.0,
8      "lowest": 277170000.0
9    },
10   "SMA_5": {
11     "avg": 277150965.8181818,
12     "highest": 277502506.0,
13     "lowest": 276346944.0
14   },
15   "EMA_10": {
16     "avg": 277018267.9226054,
17     "highest": 277289505.1079296,
18     "lowest": 276704324.598024
19   },
20   "RSI_10": {
21     "avg": 65.07231402887675,
22     "highest": 72.10420259192908,
23     "lowest": 50.96143555559214
24   },
25   "gain_loss": {
26     "highest_gain": 1734569.0,
27     "highest_loss": 569950.0
28   }
29 }
```

Summarize Multiple API

لیستی از چند نماد سهام را به عنوان ورودی گرفته و خلاصه عملکرد را برای هر کدام در بازه زمانی مشخص تولید می‌کند.

Visualization Service (Grafana)

در سرویس گرافانا که برای نمایش نمودارها و چارت‌ها استفاده شده است این قابلیت وجود دارد که به صورت socket-based برای دریافت سیگنال و همچنین query-based برای انجام عملیات aggregation استفاده شود.

در این سرویس در گرافانا یک داشبورد برای هر سهام ساخته شده است که در یک نگاه خلاصه‌ای از عملکرد سهام و نمودارهای آن نشان داده شده است که به شرح زیر است:

