

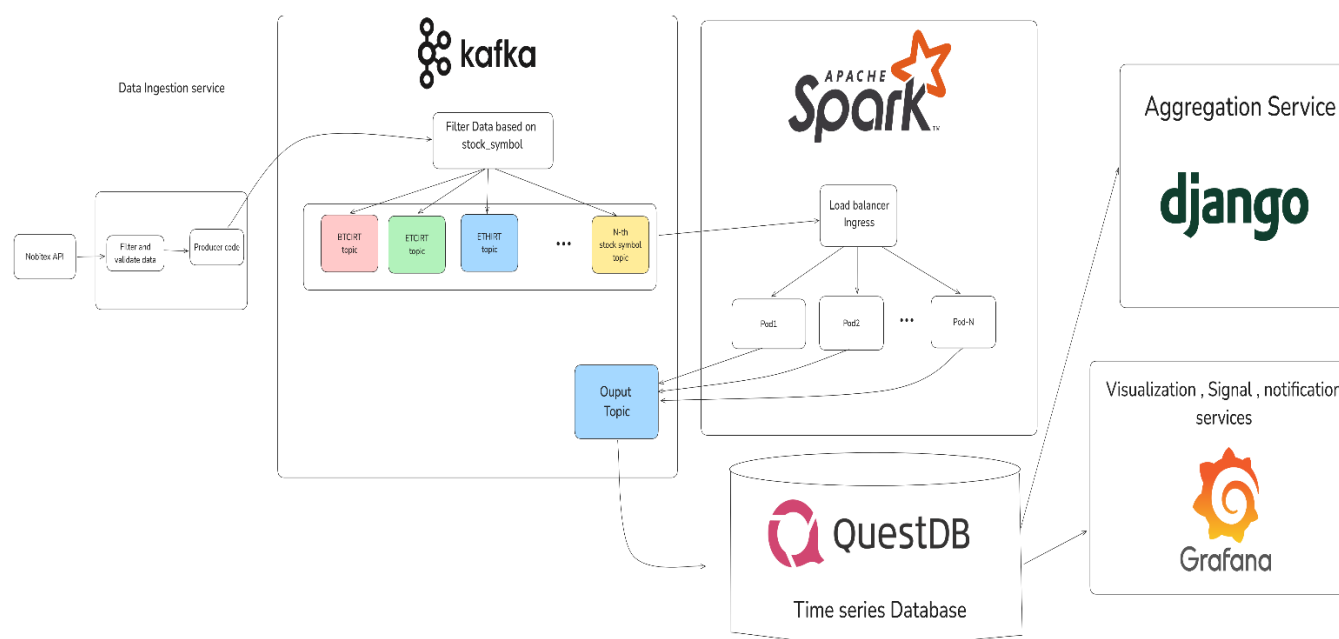
گزارش کار پروژه پایانی درس سیستم های توضیع شده

استاد درس : دکتر محسن شریفی

اعضای گروه:

اهورا امینی ، مهدی شکاری سریزدی

ساختار پروژه :

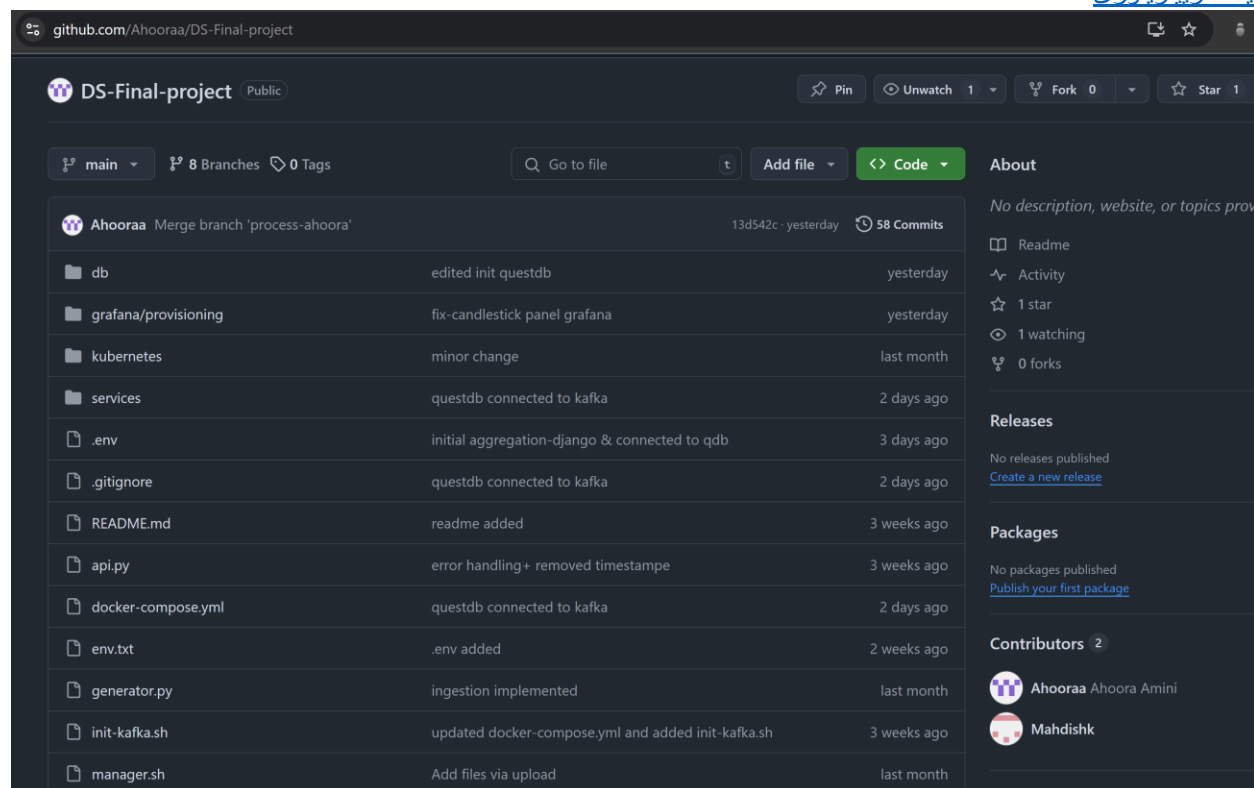


- ✓ Testing on real-world dataset and API – Nobitex API
- ✓ Data ingestion service (Kafka)
- ✓ Stream-processing service + signal generation (Using Apache Spark)
- ✓ Orchestration + Load balancing -> implemented by both Docker-compose and Minikube
- ✓ Database : Questdb

- Optimized for time-series data ,especially stock market data
- High write throughput
- Efficient time-based queries
- Real-time analytics
- Integration with stream processing frameworks such as Apache Kafka
- ✓ Aggregation service (implemented by both Django API and Grafana)
- ✓ Visualization service- Grafana (real-time visualization, auto refreshing charts, candlesticks, bar and line charts)
- ✓ Notification service- Grafana (real-time representation of stock signals using Grafana panel + Grafana alert for sending notification to apps such as slack)

روند انجام کار در گیت هاب

[لینک ریپازیتوری](#)



گرفتن قیمت سهام ها از OHLC API نوبیتکس

برای توضیحات بیشتر در مورد OHLC به [این لینک](#) مراجعه فرمایید.

• درخواست : GET /market/udf/history

- پارامترهای ورودی :

پارامتر	نوع	پیش فرض	توضیحات	نمونه
symbol	string	الزامی	نماد بازار	BTCIRT
resolution	string	الزامی	بازه زمانی هر کندل	D
to	int	الزامی	زمان پایان بازه	1562230967
from	int	اختیاری	زمان ابتدای بازه	1562058167
countback	int	اختیاری	تعداد کندل‌های پیش از زمان پایان (اولویت آن از from بیشتر است)	4
page	int	1	شماره صفحه	3

- نتیجه فراخوانی:

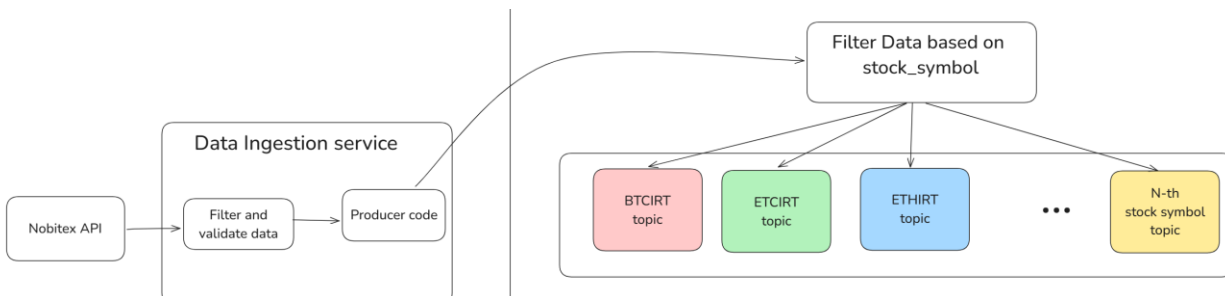
در صورت فراخوانی درست، پاسخ به این صورت خواهد بود:

```
{
  "s": "ok",
  "t": [1562095800, 1562182200],
  "o": [146272500, 150551000],
  "h": [155869600, 161869500],
  "l": [140062400, 150551000],
  "c": [151440200, 157000000],
  "v": [18.221362316, 9.8592626506]
}
```

که شامل قیمت های open, high, low, close و همچنین volume می باشد.
سپس توسط سرویس ingestion دریافت می شود.

سرویس Ingestion :

در این سرویس داده ها اعتبار سنجی می شوند و در صورت صحت داده ها ، توسط کد producer هر داده جدید بر اساس فیلد stock_symbol به یک تاپیک کافکا متناظر با آن ارسال می شود.
برای مثال داده های مربوط به سهام BTCIRT به تاپیک btcirt_topic ارسال می شود



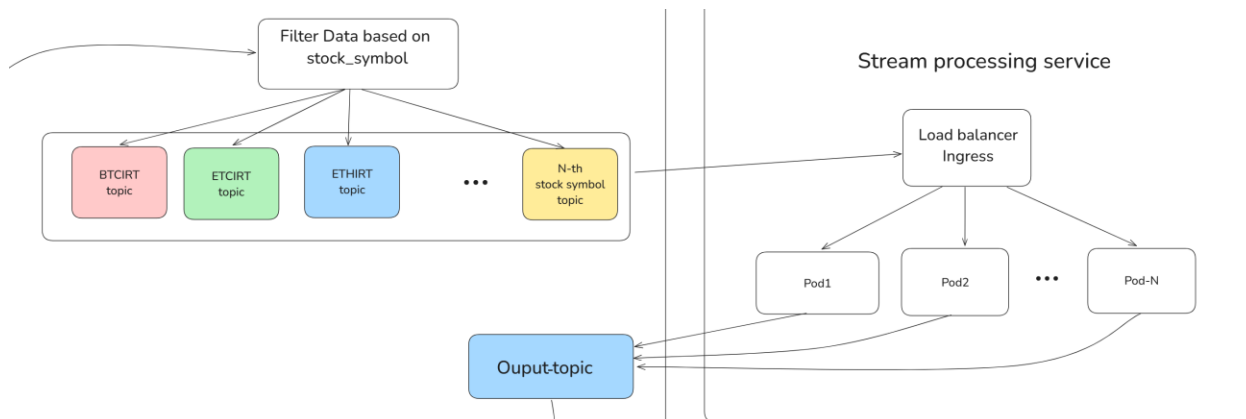
```

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': 86
97888889.0, '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, 'c
lose': 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': 2771
70000.0, '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

```

سرویس Stream processing:

این سرویس شامل چندین بخش است که به طور کلی برای پردازش داده‌های استریم از Kafka، محاسبه شاخص‌ها (SMA, EMA, RSI) ارسال داده‌ها به Kafka و مدیریت میکرو-بج‌ها با Spark طراحی شده است. توضیح بخش‌های مختلف به صورت مختصر در زیر آورده شده است:



محاسبه شاخص‌ها

- `compute_indicators_for_group` برای هر گروه از داده‌ها شاخص‌های زیر را محاسبه می‌کند:
 - **SMA(5)**: میانگین ساده ۵ دوره‌های ۵ دقیقه‌ای.

- **EMA(10)**: میانگین نمایی دوره های ۱۰ دقیقه ای
- **RSI(10)**: شاخص قدرت نسبی دوره های ۱۰ دقیقه ای.
- **Average gain & Average loss** در دوره های ۱۰ دقیقه ای

تولید سیگنال‌ها

- `generate_signals_scenario_b`: سیگنال‌های خرید، فروش، یا نگه داشتن را بر اساس شرایط زیر تولید می‌کند:

○ **خرید** $RSI < 70$ و $SMA(5) > EMA(10)$

○ **فروش** $RSI > 30$ و $SMA(5) < EMA(10)$

○ در غیر این صورت: **نگه داشتن**.

پردازش میکرو-بچ‌ها

- `process_batch`: هر میکرو-بچ 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 (مثل `stock_symbol`, `local_time`, `close`, ...) تعریف شده است

مدیریت استریم

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

```
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 explicitl
y 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
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 explicitl
y 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
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 explicitl
y 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
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 explicitl
y 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
```

سرویس میانی Kafka-to-questdb

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

```
kafka-to-questdb
c70baeb62ee3 ds-final-project-kafka-to-questdb:latest (was ds-final-project-kafka-to-questdb:<none>)
STATUS
Running (11 minutes ago)

Logs Inspect Bind mounts Exec Files Stats

volume=0,SMA_5=8685384379,EMA_10=NaN,delta=111111,gain=111111,loss=0,avg_gain_10=NaN,avg_loss_10=NaN,rs=NaN,RSI_10=NaN,signal="HOLD" 173
2025-01-25 18:38:03
2025-01-25 18:38:05 INFO:root:Received message: {"stock_symbol": "USDTIRT", "local_time": "2025-01-25 18:37:00", "open": 83554.0, "high
553.0, "close": 83559.0, "volume": 2033.3646698972, "SMA_5": 83482.4, "EMA_10": 83462.36969486231, "delta": 6.0, "gain": 6.0, "loss": -6
aN, "avg_loss_10": NaN, "rs": NaN, "RSI_10": NaN, "signal": "HOLD"}
2025-01-25 18:38:05 INFO:root:Data sent to QuestDB: stock_data,stock_symbol=USDTIRT open=83554.0,high=83590.0,low=83553.0,close=83559.0
482,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
2025-01-25 18:38:05
```

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

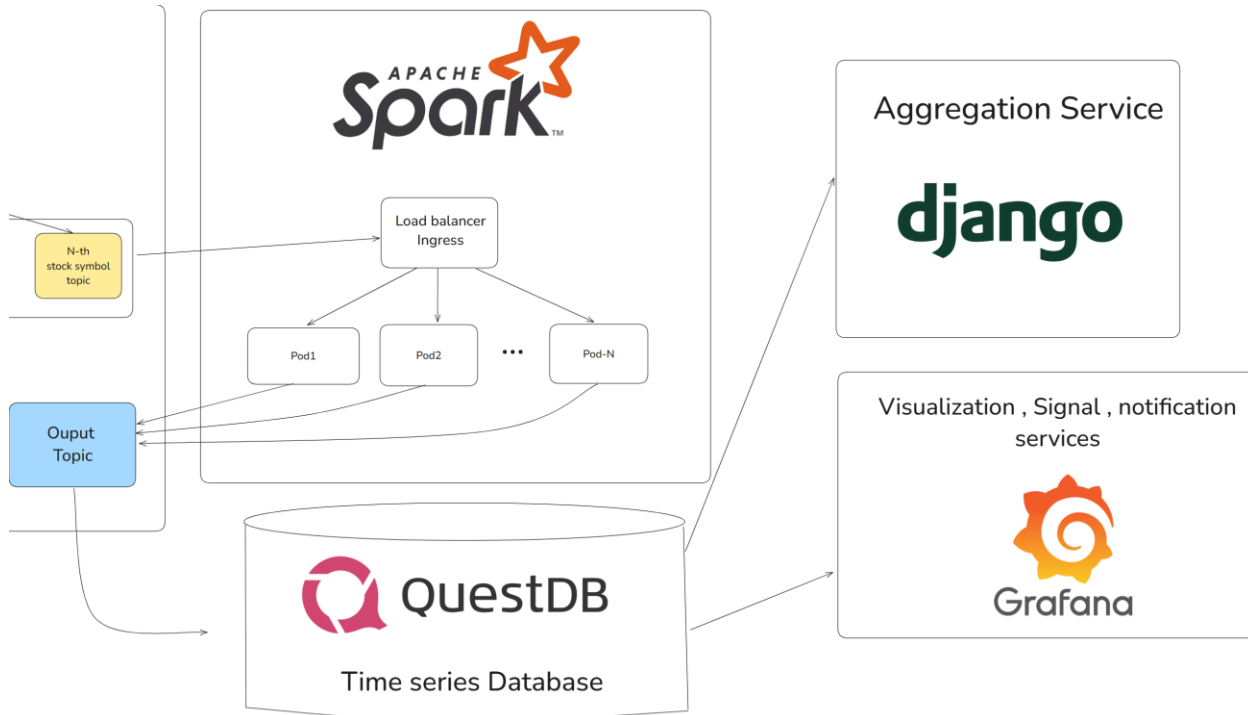
- پیام‌ها از Kafka دریافت و مقدار آن‌ها (JSON) پردازش می‌شود.
- اگر پیام شامل داده نامعتبر باشد، خطا گزارش می‌شود.

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, ...) و شاخص‌ها (RSI, SMA, ...) است



دیتابیس سری زمانی Questdb

استفاده از QuestDB برای پردازش داده‌های سری زمانی مانند داده‌های بازار سهام نسبت به ترکیب یک دیتابیس دیگر با Redis مزایای زیر را دارد:

- عملکرد بالا:
 - QuestDB برای پردازش داده‌های سری زمانی بهینه شده و می‌تواند حجم عظیمی از داده‌ها را با سرعت بسیار بالا وارد و کوئری کند

• کاهش پیچیدگی معماری :

با استفاده از QuestDB ، نیازی به ترکیب چندین ابزار) مانند دیتابیس اصلی (Redis + نیست، زیرا QuestDB به تنهایی نیازهای ذخیره سازی و تحلیل داده ها را برآورده می کند

• مقیاس پذیری

- QuestDB در مدیریت حجم عظیمی از داده های سری زمانی مقیاس پذیری بالایی دارد، در حالی که Redis برای داده های بسیار بزرگ نیاز به حافظه بیشتر و تنظیمات پیچیده تری دارد.

• پشتیبانی از: Influx Line Protocol

- امکان ارسال مستقیم داده ها با فرمت ILP باعث ساده تر شدن ارسال داده های بازار سهام می شود.

• جستجو و انجام کوئری های SQL پیشرفته:

- قابلیت استفاده از SQL استاندارد برای کوئری های پیچیده مانند محاسبه میانگین، شاخص ها (SMA, EMA)، و جستجوی مقادیر خاص در بازه های زمانی.

• پشتیبانی از داده های بلادرنگ (Realtime)

- مناسب برای سیستم های بلادرنگ که داده های بازار سهام را در لحظه پردازش و تحلیل می کنند.

• ایندکس گذاری روی زمان:

- QuestDB با استفاده از شاخص زمانی داخلی، کوئری های مبتنی بر زمان را بسیار سریع تر انجام می دهد.

• مقیاس پذیری بالا:

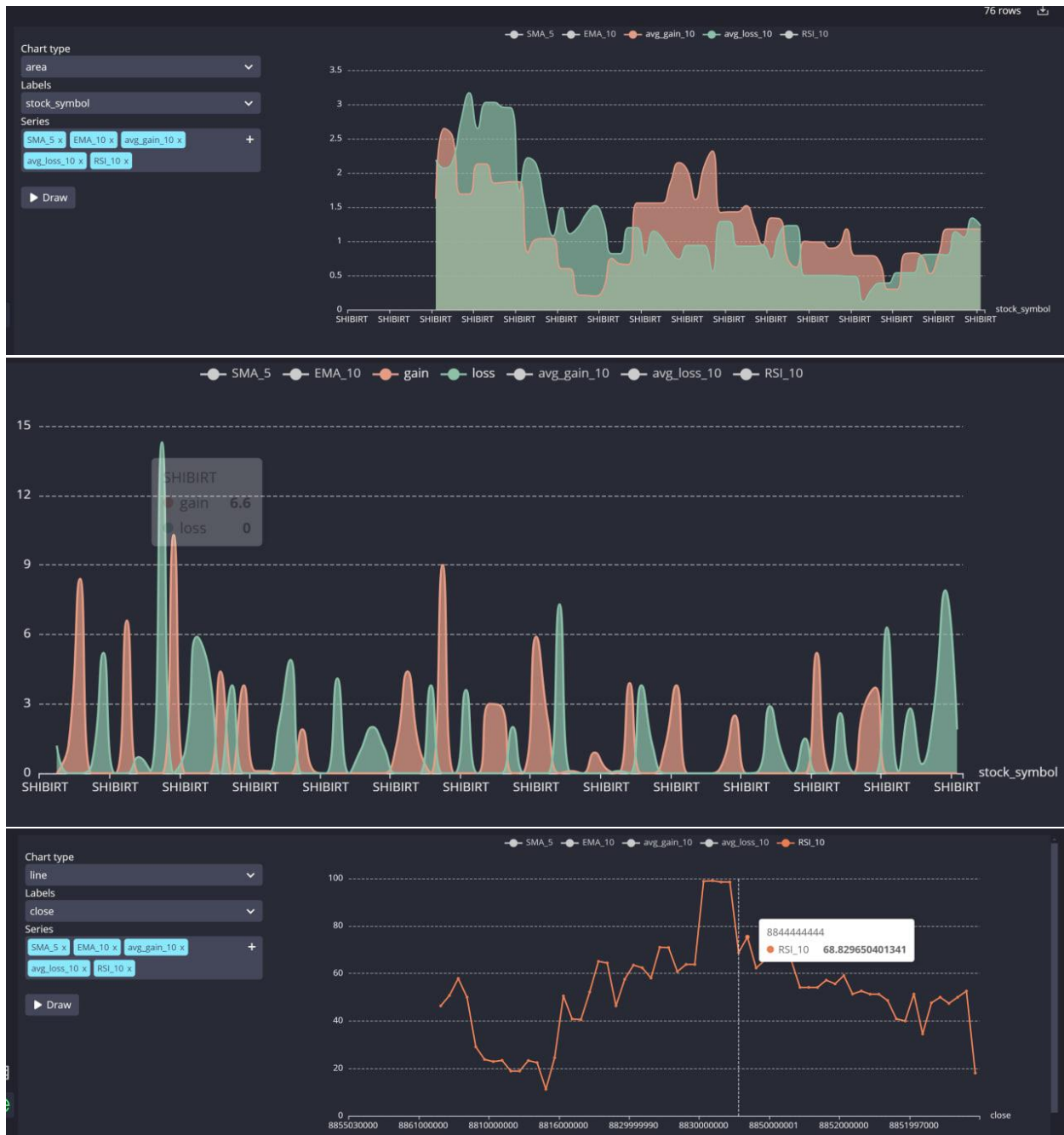
- مناسب برای ذخیره و پردازش داده های حجیم مانند داده های تاریخی بازار سهام.

تصاویری از محیط دیتابیس

2,082 rows

stock_symbol	signal	local_time	open	close	high	low	volume	SMA_5	EMA_10	delta
symbol	string	timestamp	double	double	double	double	double	double	double	double
SHIBIRT	SELL	2025-01-25T17:37:00.000000Z	1680	1680	1680	1680	91	1677	1678.679698218301	∞
BTCIRT	SELL	2025-01-25T17:38:00.000000Z	8711138043	8711138042	8711138043	8711138042	0	8711136842	8714370211.468225	-1
USDIRT	SELL	2025-01-25T17:38:00.000000Z	83766	83695	83788	83695	809	83736	83760.10839324121	-11
ETHIRT	BUY	2025-01-25T17:39:00.000000Z	278199995	278199995	278199995	278199995	0	278098468	278041127.34520024	-1
ETCIRT	BUY	2025-01-25T17:39:00.000000Z	2255017	2255017	2255017	2255017	0	2257809	2257796.158471043	1
SHIBIRT	SELL	2025-01-25T17:39:00.000000Z	1673.9	1673.9	1673.9	1673.9	0	1677	1677.81066217861	-1
BTCIRT	HOLD	2025-01-25T17:40:00.000000Z	8711138043	8711138043	8711138043	8711138043	0	8711137442	8713782544.474003	1
USDIRT	SELL	2025-01-25T17:40:00.000000Z	83712	83792	83792	83712	322	83736	83765.90686719735	95
ETHIRT	BUY	2025-01-25T17:40:00.000000Z	278199000	278199000	278199000	278199000	0	278198127	278069831.46425474	-995
ETCIRT	BUY	2025-01-25T17:40:00.000000Z	2252018	2252018	2252018	2252018	0	2257207	2256744.1296581263	-3005
SHIBIRT	SELL	2025-01-25T17:40:00.000000Z	1675.1	1675.1	1675.1	1675.1	59	1676	1677.317814509771	∞

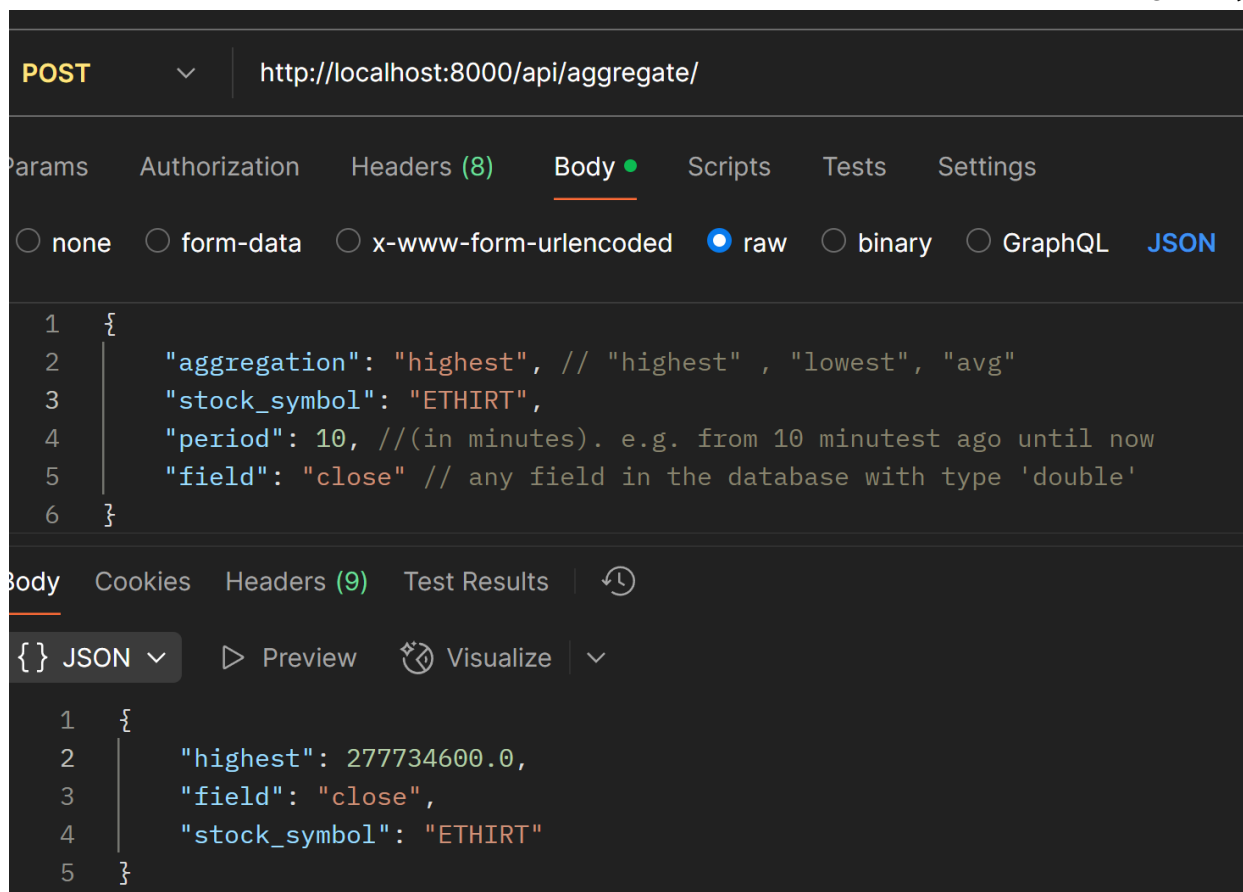
پشتیبانی از چارت های مختلف برای تحلیل بصری داده :



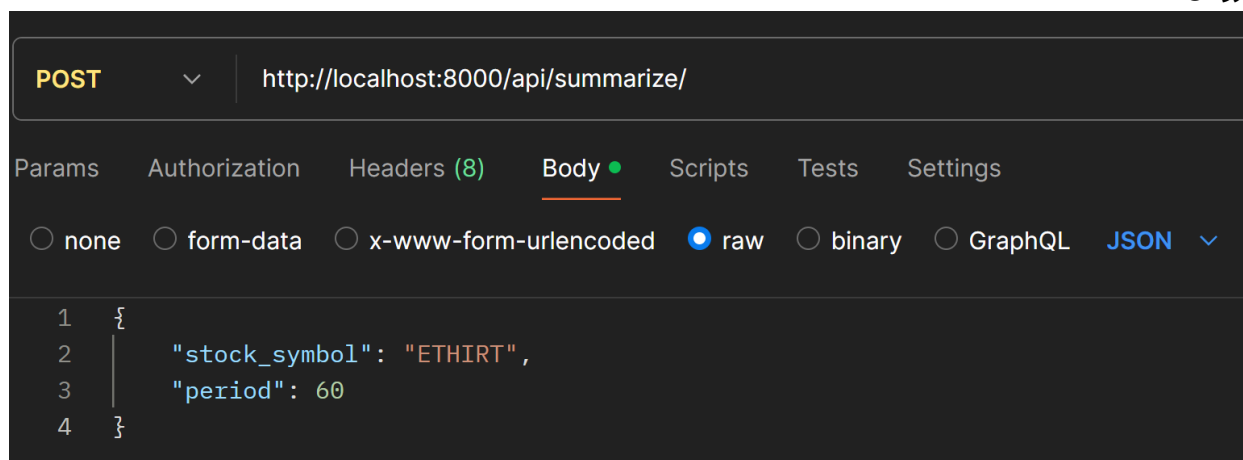
سرویس Aggregation (پیاده سازی با Django):

در این سرویس یک اپلیکیشن جنگو پیاده سازی شده که شامل ۳ نوع API هست.

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



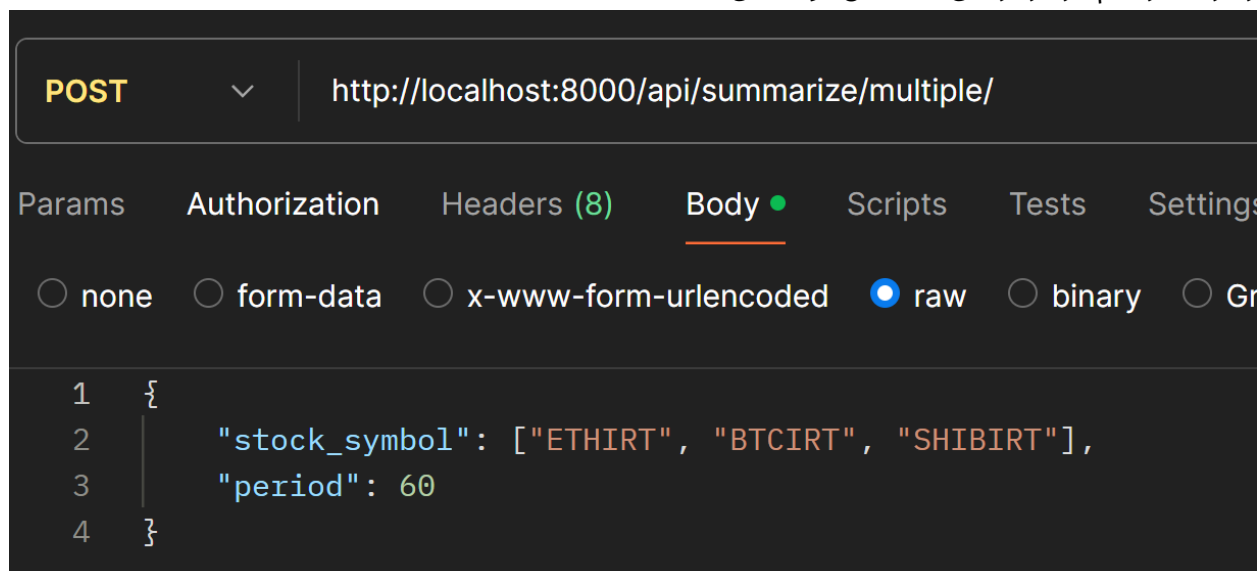
۲. خلاصه عملکرد یک سهام در بازه زمانی مشخص
این API خلاصه عملکرد سهام را برای معیار های `close, gain, loss, SMA_5, EMA_10, RSI_10` تولید می کند.
ورودی:



خروجی:

```
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         },
```

۳. Summarize multiple API یک لیستی از نماد چند سهام به عنوان ورودی می‌گیرد و خلاصه عملکرد را برای هرکدام در بازه زمانی مشخص تولید می‌کند:



The screenshot shows a REST client interface with a POST request to `http://localhost:8000/api/summarize/multiple/`. The 'Body' tab is selected, and the 'raw' radio button is chosen. The JSON body is as follows:

```
1 {  
2   "stock_symbol": ["ETHIRT", "BTCIRT", "SHIBIRT"],  
3   "period": 60  
4 }
```

خروجی :

```

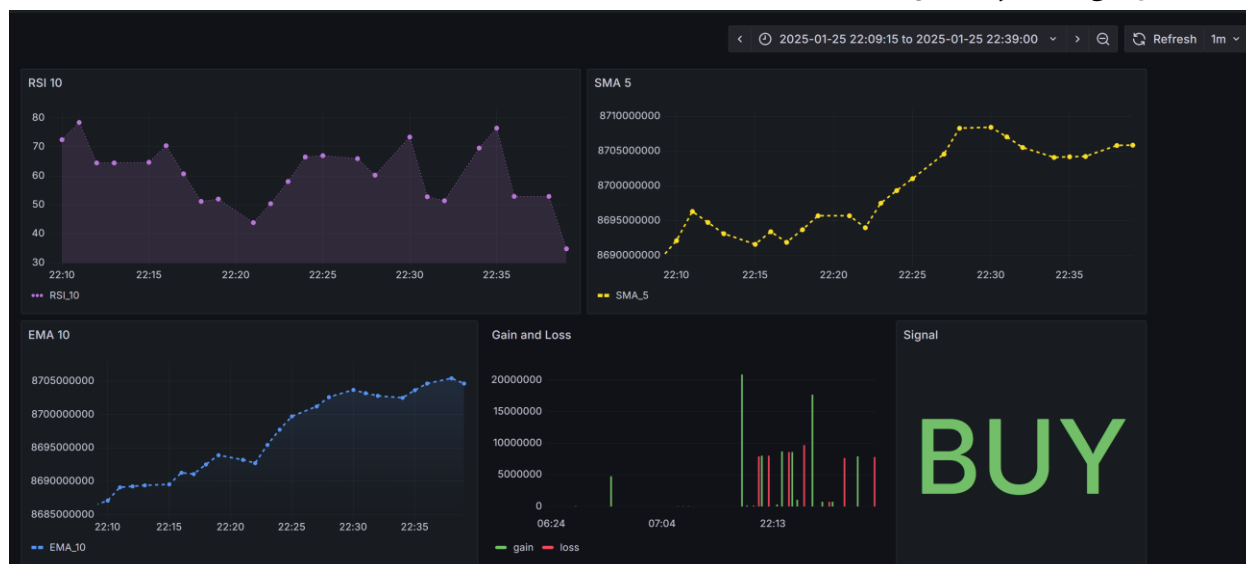
18         "lowest": 276704324.598024
19     },
20     "RSI_10": {
21         "avg": 63.82374064370655,
22         "highest": 72.10420259192908,
23         "lowest": 50.96143555559214
24     },
25     "gain_loss": {
26         "highest_gain": 1734569.0,
27         "highest_loss": 569950.0
28     }
29 },
30 "BTCIRT": {
31     "close": {
32         "avg": 8694363492.285715,
33         "highest": 8700011110.0,
34         "lowest": 8690000000.0
35     },
36     "SMA_5": {
37         "avg": 8692063317.571428,
38         "highest": 8696355555.0,
39         "lowest": 8681186580.0
40     },
41     "EMA_10": {

```

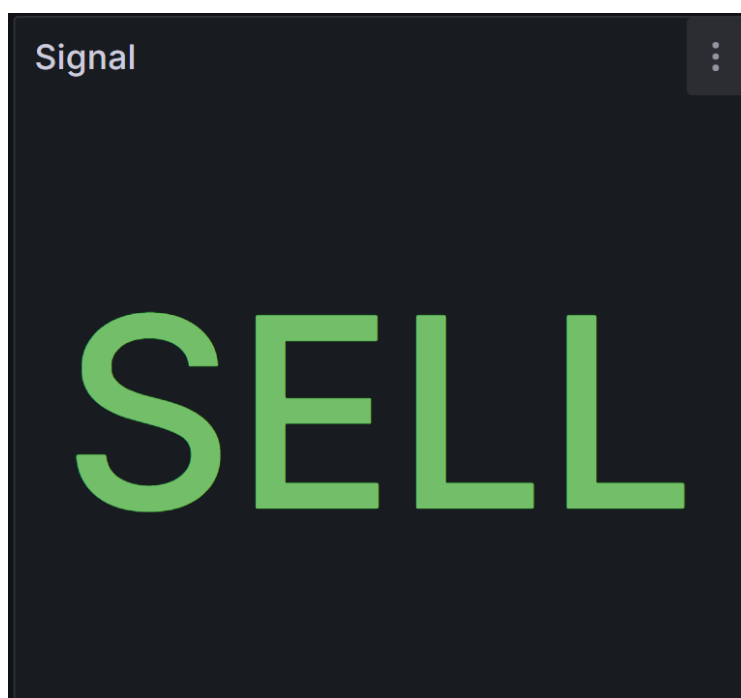
سرویس های Visualization , Aggregation, Signal (پایاده سازی شده با Grafana) Notificaton

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

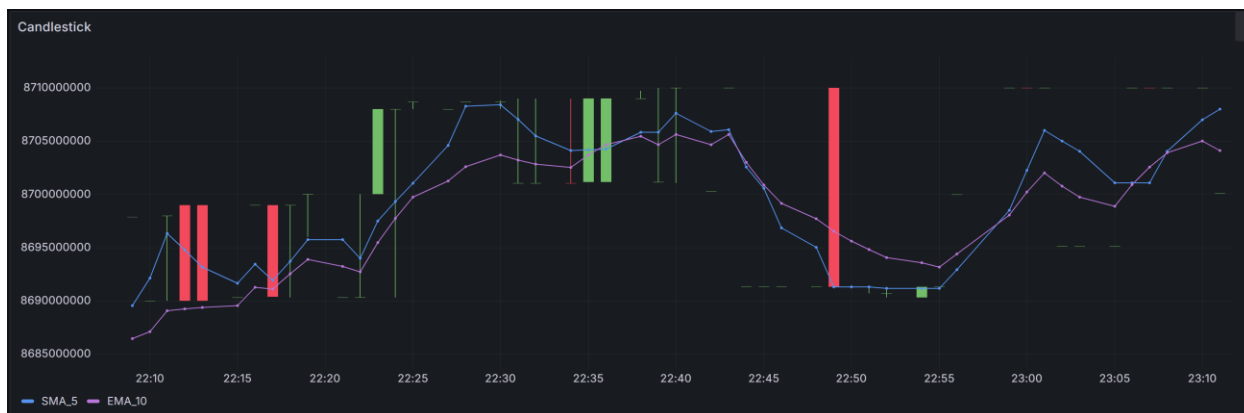
ما در گرافانا یک داشبورد برای هر سهام ساخته ایم که در یک نگاه خلاصه ای از عملکرد سهام و نمودارهای آنرا به ما نشان می‌دهد. برای مثال:



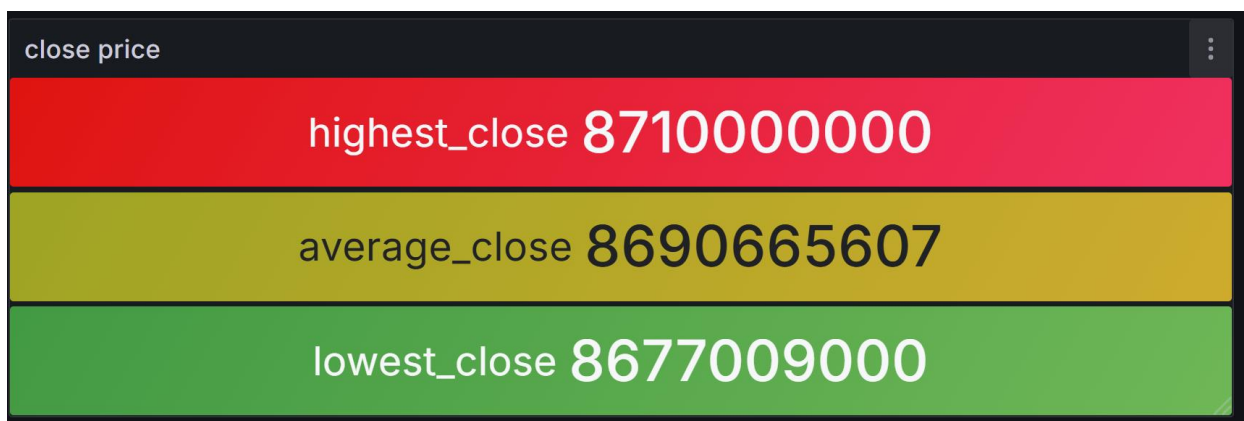
قابلیت نشان دادن سیگنال به صورت کوئری و یا استفاده از websocket



در نمودار بعدی ما با استفاده از مقادیر دریافتی سهام candlestick آنرا در گرافانا ساخته ایم و مقادیر EMA, SMA را برای تحلیل و آنالیز سهام نشان داده ایم



(خط بنفش نشان دهنده EMA و خط آبی نشان دهنده SMA) می باشد.
و همچنین اعمال عملیات aggregation برای نمایش خلاصه ای از عملکرد سهام



Orchestrate کردن پروژه با Minikube

در این قسمت به توضیح معماری و کدهای پیاده سازی شده برای استقرار سرویس های پردازش داده در یک خوشه Minikube می پردازد. سیستم طراحی شده شامل سرویس های Kafka ، QuestDB ، Spark و سرویس های مرتبط است که با همکاری یکدیگر، داده های مالی را جمع آوری، پردازش و ذخیره می کنند.

فایل kafka_service.yaml :

هدف: ایجاد یک بروکر Kafka برای مدیریت جریان داده

استفاده از image `bitnami/kafka:3.5.1` با پشتیبانی از KRaft نسخه بدون Zookeeper .

```

- name: KAFKA_CFG_ADVERTISED_LISTENERS
  value: "PLAINTEXT://kafka:9092"
- name: KAFKA_CFG_BROKER_ID
  value: "1"
- name: KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR
  value: "1"
- name: KAFKA_CFG_LOG_DIRS
  value: "/opt/kafka-logs"
- name: KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE
  value: "false"
- name: KAFKA_CFG_PROCESS_ROLES
  value: "broker,controller" # است KRaft این خط برای فعال کردن
- name: KAFKA_CFG_CONTROLLER_LISTENERS
  value: "PLAINTEXT://0.0.0.0:9093"
- name: KAFKA_CFG_CONTROLLER_QUORUM_VOTERS
  value: "1@localhost:9093"
- name: KAFKA_CFG_NODE_ID
  value: "1"
- name: KAFKA_CFG_CONTROLLER_LISTENER_NAMES
  value: "CONTROLLER"

```

ایجاد سرویس در پورت ۹۰۹۲:

```

apiVersion: v1
kind: Service
metadata:
  name: kafka
spec:
  selector:
    app: kafka
  ports:
    - protocol: TCP
      port: 9092
      targetPort: 9092
  type: ClusterIP

```

سرویس QuestDB :

هدف: پایگاه داده سریع برای ذخیره داده‌های زمان‌محور

در عکس‌های بعدی تنظیمات تعریف پورت‌های ضروری آمده است.

```

ports:
- name: http
  protocol: TCP
  port: 9000 You, 2 days ago • add kubernetes kafka and questdb services
  targetPort: 9000
  nodePort: 30090
- name: tcp
  protocol: TCP
  port: 8812
  targetPort: 8812
  nodePort: 30091
- name: ilp
  protocol: TCP
  port: 9009
  targetPort: 9009
  nodePort: 30092 # Optional for external access

```

سرویس Spark :

هدف: پردازش جریان داده با استفاده از Apache Spark .

سرویس spark master :

```
kind: Service
apiVersion: v1
metadata:
  name: spark-master
spec:
  ports:
    - name: webui
      port: 8080
      targetPort: 8080
    - name: spark
      port: 7077
      targetPort: 7077
  selector:
    component: spark-master
```

پاد spark worker که مقیاس پذیر است و می‌تواند رپلیکای بیشتری داشته باشد:

```

kind: Deployment
apiVersion: apps/v1
metadata:
  name: spark-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      component: spark-worker
  template:
    metadata:
      labels:
        component: spark-worker
    spec:
      containers:
        - name: spark-worker
          image: spark-hadoop:3.2.0
          imagePullPolicy: Never
          command: ["/spark-worker"]
          ports:
            - containerPort: 8081
          resources:
            requests:
              cpu: "1"
              memory: "2Gi"
            limits:
              cpu: "2"
              memory: "4Gi"

```

و قسمت consumer که تحلیل داده ها را برعهده دارد .

فایل kafka_to_questdb_service.yaml :

هدف: انتقال داده از Kafka به QuestDB .

که کد فایل yaml آن به شکل زیر است:

```

1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: kafka-consumer
5  spec:
6    containers:
7      - name: consumer
8        image: kafka-consumer:latest # Must match your built image name
9        imagePullPolicy: Never # For local Minikube images
10       env:
11         - name: PYTHONUNBUFFERED
12         value: "1"

```

سرویس Ingestor :

هدف: تزریق داده به Kafka .

قسمت مهم فایل yaml آن:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: data-ingestion-service
5    labels:
6      app: data-ingestion
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: data-ingestion
12   template:
13     metadata:
14       labels:
15         app: data-ingestion
16     spec:
17       containers:
18       - name: data-ingestion-service
19         image: data-ingestion-service:4
20         imagePullPolicy: IfNotPresent
21         env:
22         - name: KAFKA_BROKER
23           value: "kafka:9092" # Match your Kafka service name in Minikube
24         ports:
25         - containerPort: 5000
26
```

حال با زدن دستور زیر هر component را خواهیم ساخت:

Kubectl apply -f <file_servic.yaml>

که با دستور pod get kubectl پاد های ساخته شده را میبینیم:

NAME	READY	STATUS	RESTARTS	AGE
data-ingestion-service-69f8bbb9f4-5jg4x	1/1	Running	9 (157m ago)	19h
grafana-64948ff7f5-6m9cm	1/1	Running	0	85s
kafka-64664889fb-j9s8m	1/1	Running	2 (5h5m ago)	21h
kafka-consumer	1/1	Running	0	110m
questdb-69ff4fcfc4-tnnbt	1/1	Running	1 (116m ago)	116m
spark-consumer	1/1	Running	0	152m
spark-master-d89f64bf6-cmd9p	1/1	Running	2 (5h5m ago)	20h
spark-worker-5cd66c7477-6wk2s	1/1	Running	3 (157m ago)	7h45m

که همانطور که مشاهده میکنید تمامی پادها سالم و درست در حال اجرا هستند.

حال با دستور `kubectl get services` سرویس های ساخته شده را مشاهده میکنیم.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
grafana-service	NodePort	10.106.58.217	<none>	3000:30300/TCP	118s
kafka	ClusterIP	10.103.172.164	<none>	9092/TCP	21h
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	21h
questdb-service	NodePort	10.100.179.54	<none>	9000:30090/TCP,8812:30091/TCP,9009:30092/TCP	21h
spark-master	ClusterIP	10.110.79.166	<none>	8080/TCP,7077/TCP	20h

حال با رفتن در هر پاد از صحت کارکرد کلاستر اطمینان حاصل میکنیم.
کارکرد درست ingestion service با چک کردن دیتا یک تایپیک:

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic btcirt_topic --from-beginning
```

```
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:26:00\", \"open\": 8702990001.0, \"high\": 8703000000.0, \"low\": 8702990001.0, \"close\": 8702990001.0, \"volume\": 0.0007856908, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:27:00\", \"open\": 8703000000.0, \"high\": 8703000000.0, \"low\": 8702990001.0, \"close\": 8703000000.0, \"volume\": 0.040495595, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:28:00\", \"open\": 8703000000.0, \"high\": 8703000000.0, \"low\": 8703000000.0, \"close\": 8703000000.0, \"volume\": 0.000344, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:29:00\", \"open\": 8704898555.0, \"high\": 8704898555.0, \"low\": 8704898555.0, \"close\": 8704898555.0, \"volume\": 0.004180869, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:30:00\", \"open\": 8707995000.0, \"high\": 8707995000.0, \"low\": 8707995000.0, \"close\": 8707995000.0, \"volume\": 0.002880446, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:31:00\", \"open\": 8704898555.0, \"high\": 8707995000.0, \"low\": 8704898555.0, \"close\": 8707995000.0, \"volume\": 0.003875, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:32:00\", \"open\": 8707995000.0, \"high\": 8707995000.0, \"low\": 8707995000.0, \"close\": 8707995000.0, \"volume\": 0.001259, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:34:00\", \"open\": 8704898555.0, \"high\": 8704898555.0, \"low\": 8704898555.0, \"close\": 8704898555.0, \"volume\": 8.825e-07, \"topic\": \"btcirt_topic\"}
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:35:00\", \"open\": 8707995000.0, \"high\": 8707995000.0, \"low\": 8707995000.0, \"close\": 8707995000.0, \"volume\": 5.7e-05, \"topic\": \"btcirt_topic\"}
```

در قسمت بعد برای چک کردن کارکرد stream processing تایپیک output_topic را میخوانیم:

```
{\"stock_symbol\": \"BTCIRT\", \"local_time\": \"2025-01-25 22:47:00\", \"open\": 8707995000.0, \"high\": 8707995000.0, \"low\": 8707995000.0, \"close\": 8707995000.0, \"volume\": 3.95e-07, \"SMA_5\": 8708396000.0, \"EMA_10\": 8707757643.036777, \"delta\": 0.0, \"gain\": 0.0, \"loss\": -0.0, \"avg_gain_10\": 809644.5, \"avg_loss_10\": 809644.5, \"rs\": 1.0, \"RSI_10\": 50.0, \"signal\": \"BUY\"}
{\"stock_symbol\": \"USDTIIRT\", \"local_time\": \"2025-01-25 22:47:00\", \"open\": 83516.0, \"high\": 83545.0, \"low\": 83516.0, \"close\": 83545.0, \"volume\": 491.95, \"SMA_5\": 83512.4, \"EMA_10\": 83513.93685439046, \"delta\": 43.0, \"gain\": 43.0, \"loss\": -0.0, \"avg_gain_10\": 5.9, \"avg_loss_10\": 2.8, \"rs\": 2.107142857142857, \"RSI_10\": 67.816091954023, \"signal\": \"SELL\"}
{\"stock_symbol\": \"ETHIRT\", \"local_time\": \"2025-01-25 22:47:00\", \"open\": 278653404.0, \"high\": 278653404.0, \"low\": 278653404.0, \"close\": 278653404.0, \"volume\": 0.0, \"SMA_5\": 278671642.2, \"EMA_10\": 278706317.6847721, \"delta\": 0.0, \"gain\": 0.0, \"loss\": -0.0, \"avg_gain_10\": 9117.0, \"avg_loss_10\": 43775.4, \"rs\": 0.2082676571731877, \"RSI_10\": 17.236880912947797, \"signal\": \"HOLD\"}
{\"stock_symbol\": \"ETCIRT\", \"local_time\": \"2025-01-25 22:47:00\", \"open\": 2264227.0, \"high\": 2269000.0, \"low\": 2264227.0, \"close\": 2269000.0, \"volume\": 10.461605, \"SMA_5\": 2264354.0, \"EMA_10\": 2264334.239752821, \"delta\": 5784.0, \"gain\": 5784.0, \"loss\": -0.0, \"avg_gain_10\": 1689.3, \"avg_loss_10\": 582.8, \"rs\": 2.8985929993136583, \"RSI_10\": 74.34972052286432, \"signal\": \"HOLD\"}
{\"stock_symbol\": \"SHIBIRT\", \"local_time\": \"2025-01-25 22:47:00\", \"open\": 1674.1, \"high\": 1674.1, \"low\": 1674.1, \"close\": 1674.1, \"volume\": 0.0, \"SMA_5\": 1676.3799999999999, \"EMA_10\": 1675.9107973588107, \"delta\": 0.09999999999999905, \"gain\": 0.09999999999999905, \"loss\": -0.0, \"avg_gain_10\": 1.6400000000000009, \"avg_loss_10\": 1.43000000000000181, \"rs\": 1.1468531468531387, \"RSI_10\": 53.42019543973924, \"signal\": \"BUY\"}
```

سپس با دستور زیر برای چک کردن questDB اقدام میکنیم:

```
(base) mahdi@mahdi:~/Uni_Project/DS-Final-project$ kubectl port-forward svc/questdb-service 9000:9000
Forwarding from 127.0.0.1:9000 -> 9000
Forwarding from [::1]:9000 -> 9000
Handling connection for 9000
Handling connection for 9000
Handling connection for 9000
```

بعد قسمت UI آن را چک می‌کنیم:

Warning: vm.max_map_count limit is too low [current=262144, recommended=1048576] Max virtual memory areas limit

SQL 1

```
1 | SELECT * FROM stock_data
```

Log [22:50:58 GMT+03:30] ✓ 3,292 rows in 18ms Execute: 2.24ms Network: 15.76ms Total: 18ms Count: 18.3µs Authentication: 4.41µs Compile: 0

3,292 rows

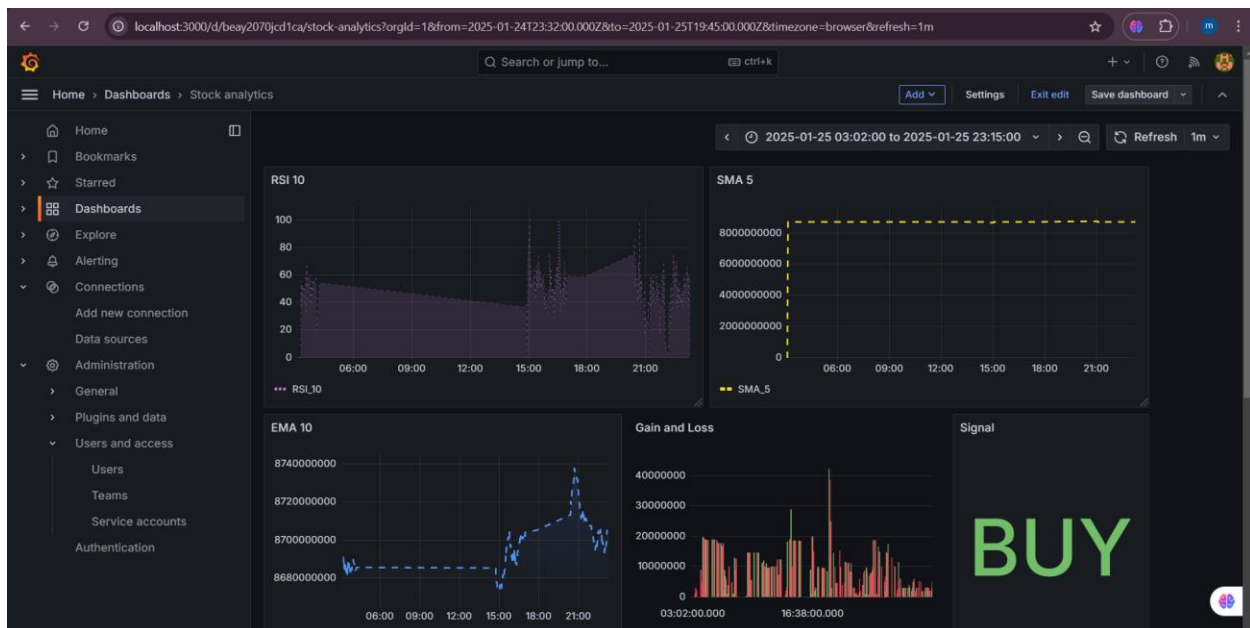
stock_symbol	signal	local_time	open	close	high	low	volume	SMA_5	EMA_10	delta
symbol	string	timestamp	double	double	double	double	double	double	double	double
SHIBIRT	BUY	2025-01-25T22:47:00.000000Z	1674.1	1674.1	1674.1	1674.1	0	1676	1675.91879735881	
BTCIRT	BUY	2025-01-25T22:48:00.000000Z	8710000000	8707995000	8710000000	8707995000	0	8708396000	8707800798.848272	
USDIRT	SELL	2025-01-25T22:48:00.000000Z	83580	83501	83580	83501	493	83509	83511.58469904673	-4
ETHIRT	HOLD	2025-01-25T22:48:00.000000Z	278653404	278653405	278653405	278653404	0	278662523	278696697.19663167	
ETCIRT	BUY	2025-01-25T22:48:00.000000Z	2269000	2269000	2269000	2269000	13	2265520	2265182.5597977624	
SHIBIRT	BUY	2025-01-25T22:48:00.000000Z	1674.1	1675	1675	1674.1	1082	1675	1675.745197839026	
BTCIRT	BUY	2025-01-25T22:49:00.000000Z	8707995001	8710000000	8710000000	8707995001	0	8708396000	8708200653.60313	2005000
USDIRT	BUY	2025-01-25T22:49:00.000000Z	83580	83510	83597	83501	1593	83511	83511.29657194731	
ETHIRT	SELL	2025-01-25T22:49:00.000000Z	278653404	278653404	278653405	278653404	0	278662523	278688825.706335	-1
ETCIRT	BUY	2025-01-25T22:49:00.000000Z	2269000	2269000	2269000	2269000	0	2266686	2265876.6398345325	
SHIBIRT	SELL	2025-01-25T22:49:00.000000Z	1678	1678	1678	1678	16895	1675	1676.155161868294	

Copyright © 2025 QuestDB

Connected QuestDB 8.2.1

سپس به خروجی grafana می‌پردازیم:

```
(base) mahdi@mahdi:~/Uni_Project/DS-Final-project$ kubectl port-forward svc/grafana-service 3000:3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
```



کوبر ما به درستی در حال کار کردن هست

با تشکر