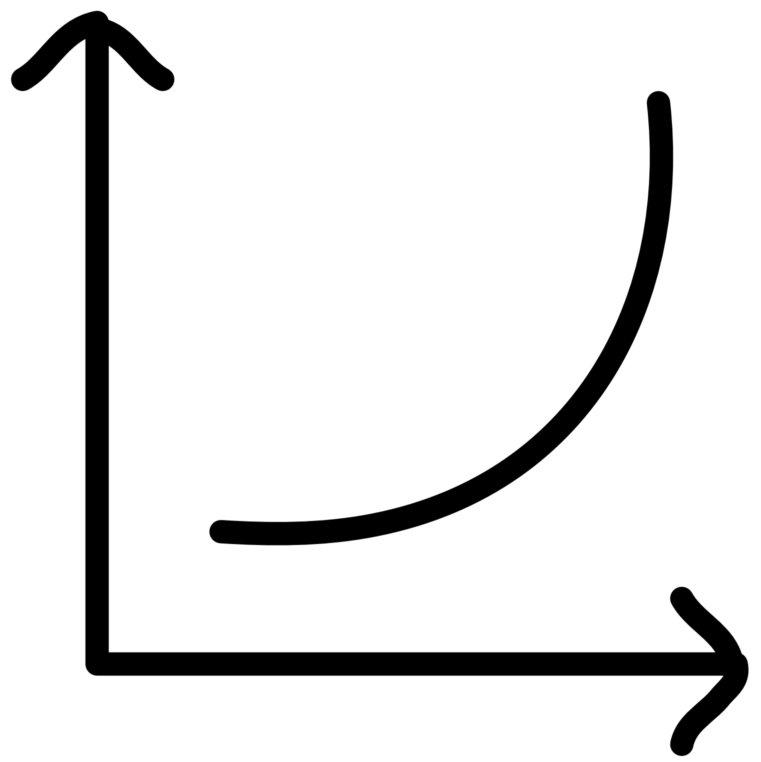
QSync

Capturing and Visualising Cryptocurrency Market Data

Sanchit Ajmera - Mazen Hussein - Mustafa Ilyas - Tyrell Duku - Luqman Liaquat
Abdur Sharif - Ayoob Ahmed

Supervised by Dr. Paul Bilokon

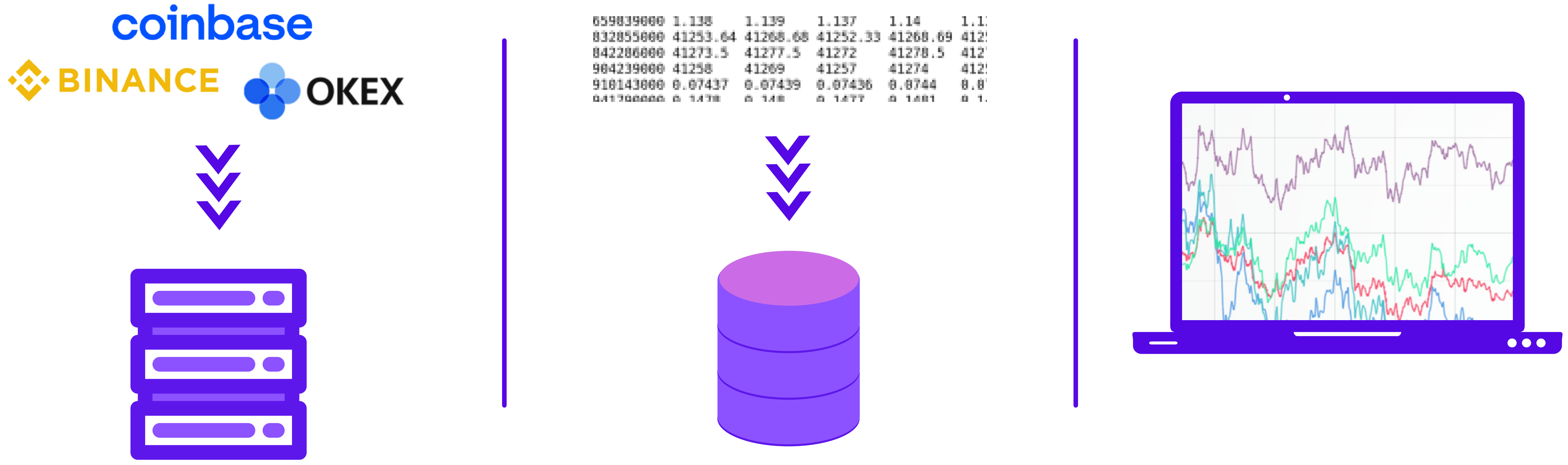
Cryptocurrency



Holders doubling in 2021 from 100 million to 200 million

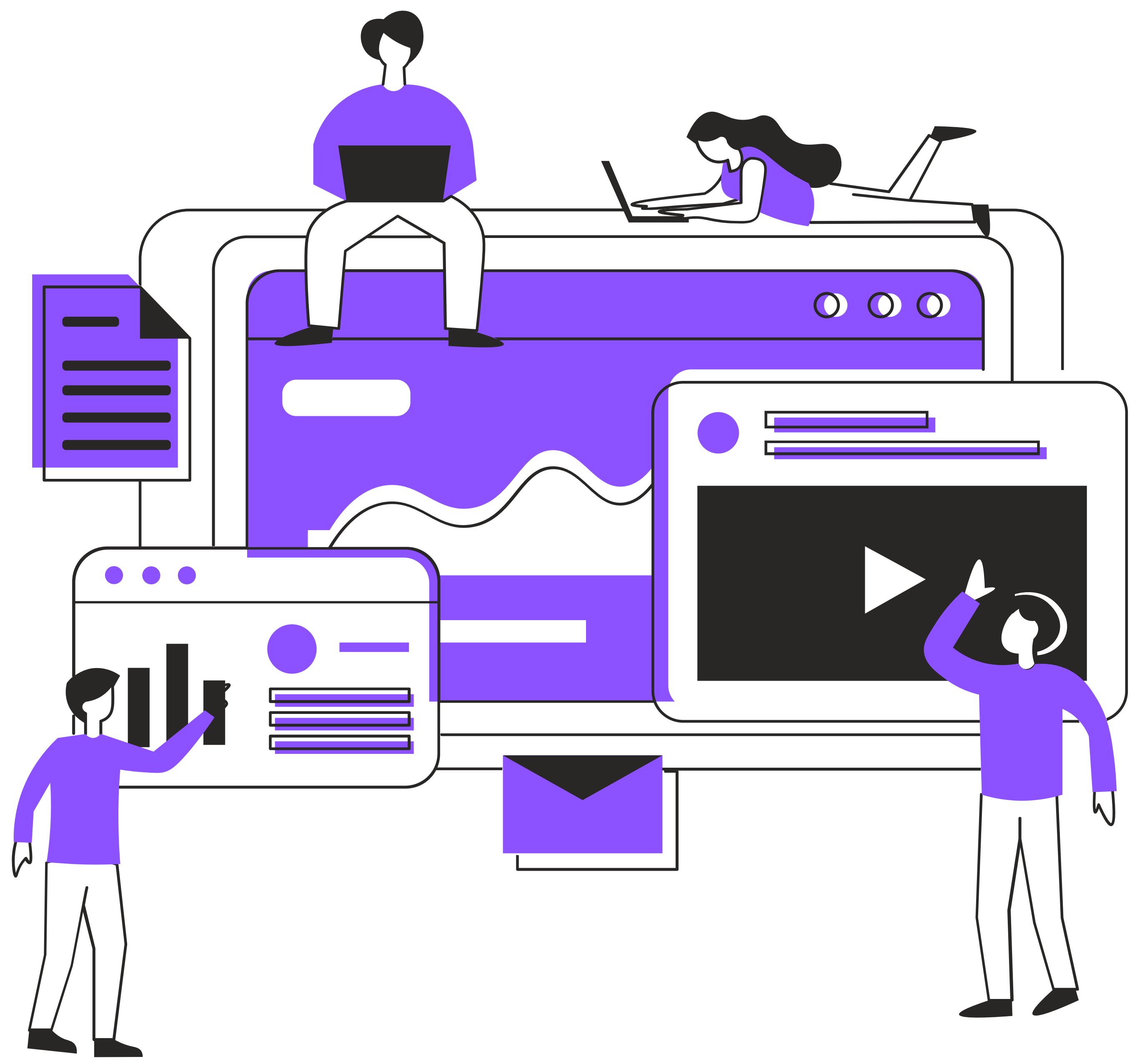
High Volatility

Our Platform

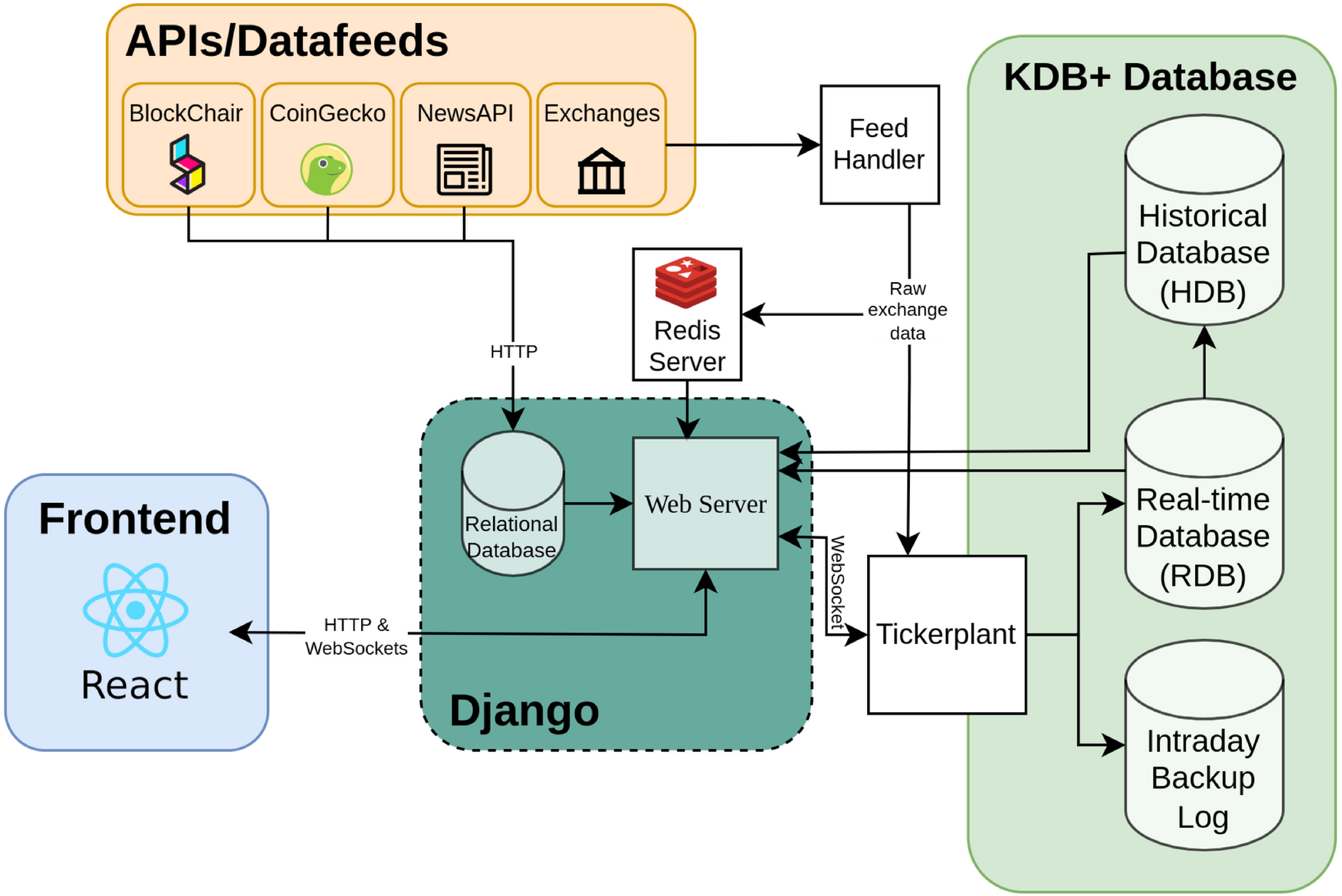


```
659839000 1.138     1.139     1.137     1.14      1.1
832855000 41253.64 41268.68 41252.33 41268.69 412
842286000 41273.5  41277.5  41272     41278.5  412
904239000 41258    41269    41257     41274    412
910143000 0.07437  0.07439  0.07436   0.0744   0.0
941290000 0.1478   0.148    0.1477    0.1481   0.1
```

Time for a Demonstration!



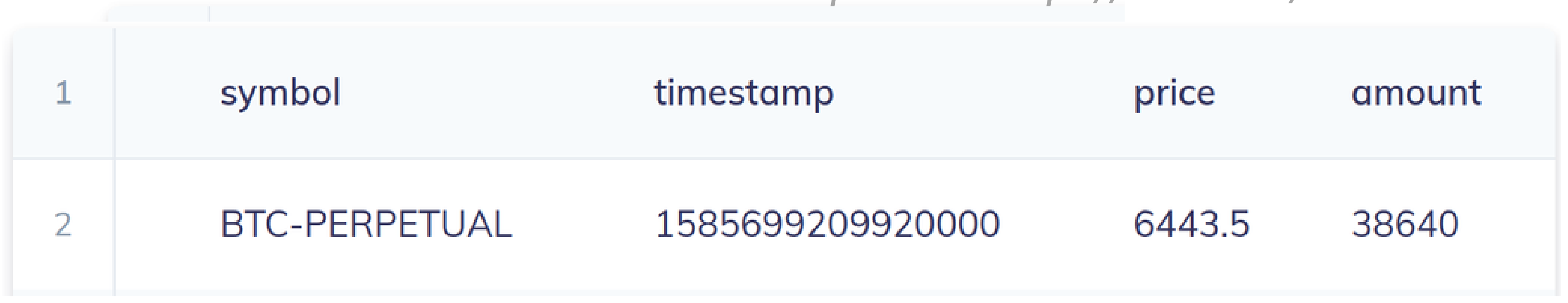
Architecture



Time-series Data

- We have **time-series** data
    - Common in any market data
    - Represent frequent & chronological events
    - Timestamped
- An example of time-series data:

Data sampled from https://tardis.dev/#csv-datasets.



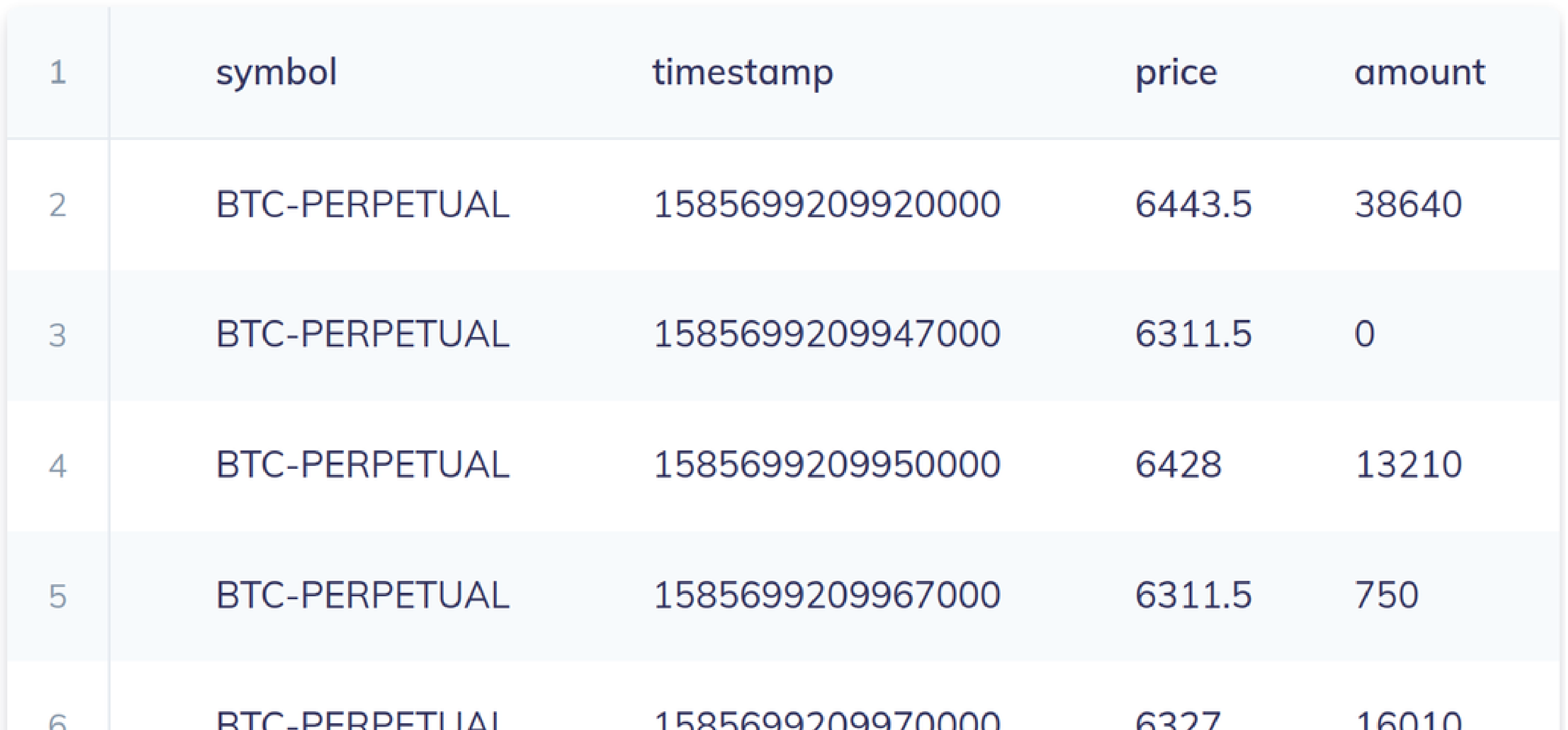
1	symbol	timestamp	price	amount
2	BTC-PERPETUAL	1585699209920000	6443.5	38640

Time-series Data

- We have **time-series** data
  - Common in any market data
  - Represent frequent & chronological events
  - Timestamped

- An example of time-series data:

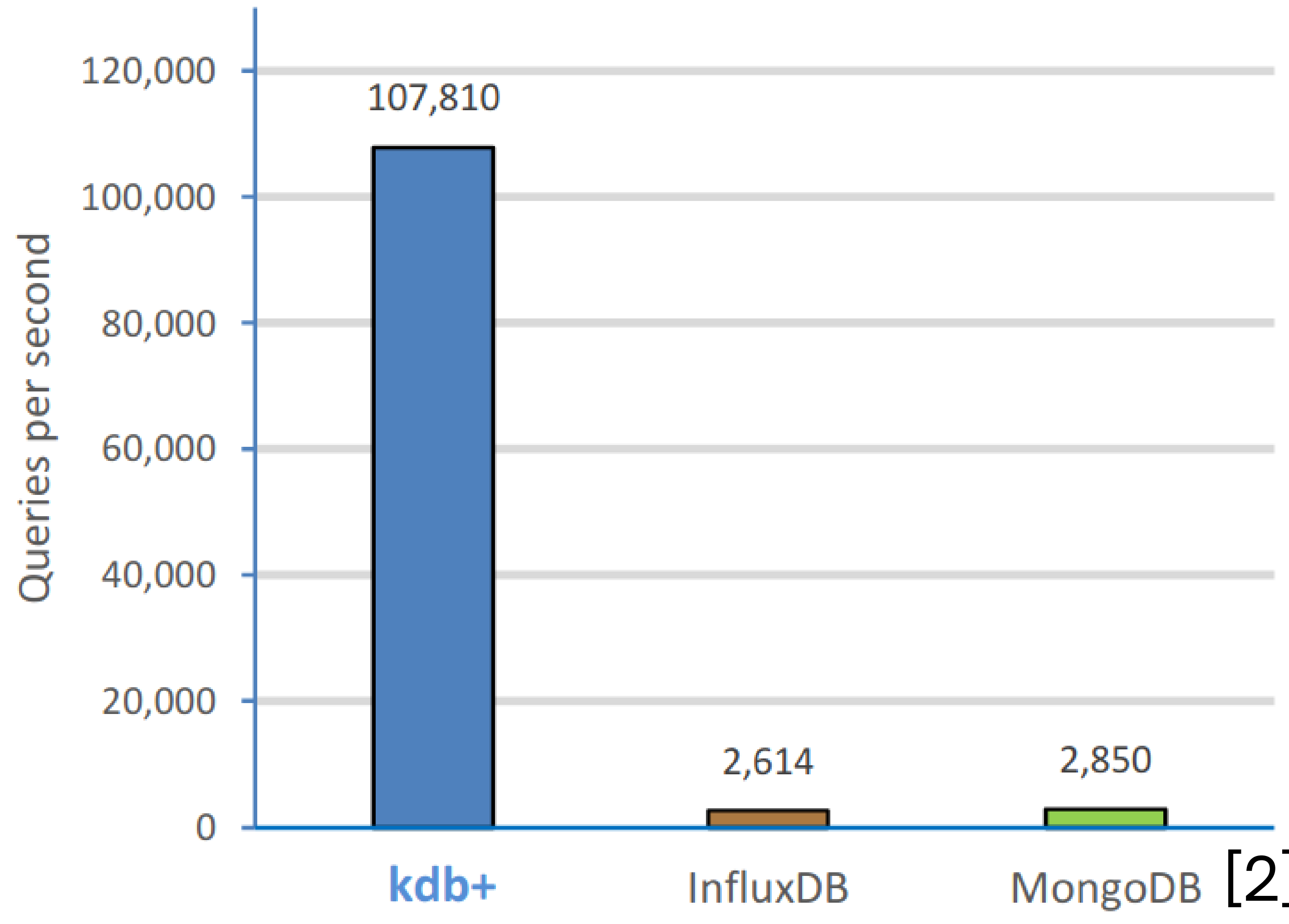
Data sampled from https://tardis.dev/#csv-datasets.



symbol	timestamp	price	amount
BTC-PERPETUAL	1585699209920000	6443.5	38640
BTC-PERPETUAL	1585699209947000	6311.5	0
BTC-PERPETUAL	1585699209950000	6428	13210
BTC-PERPETUAL	1585699209967000	6311.5	750
BTC-PERPETUAL	1585699209970000	6327	16010

The Database: kdb+

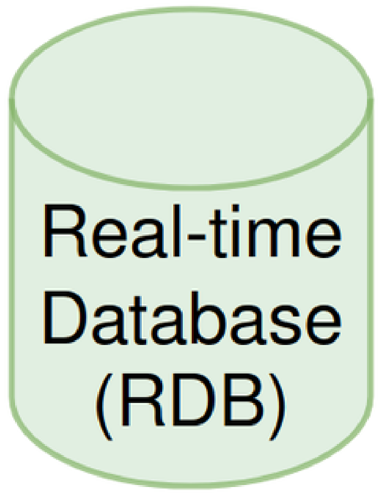
- kdb+
  - Powerful on **time-series** data
  - Proven industrial usage
  - Columnar structure



[2]

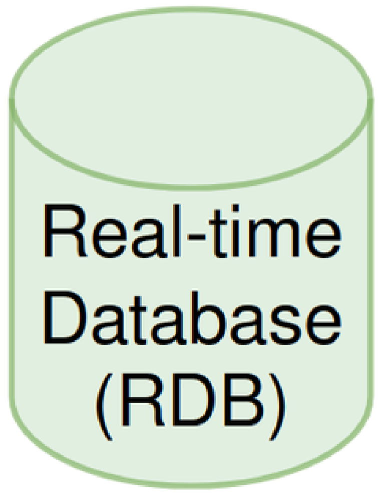
[1] Tomczak P. What Makes Time-Series Database kdb+ So Fast? [Internet]. KX. 2022 . Available from: https://kx.com/blog/what-makes-time-series-database-kdb-so-fast
[2] KDB+ transitive comparisons - KX [Internet]. 2018. Available from: https://kx.com/wp-content/uploads/2020/11/KdbTransitive-Comparisons-1.pdf

kdb+tick architecture

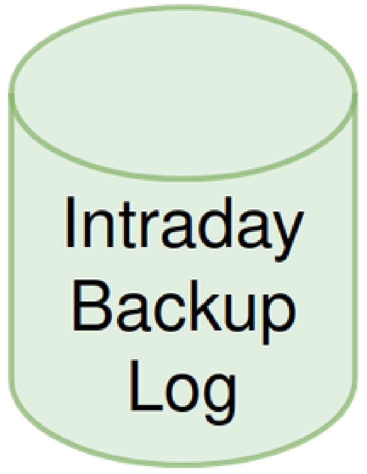


- in-memory
- today's data **only**

kdb+tick architecture

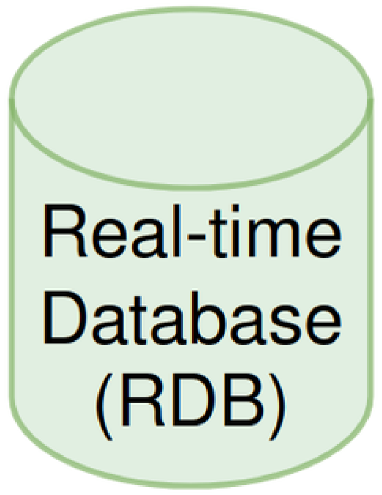


- in-memory
- today's data **only**

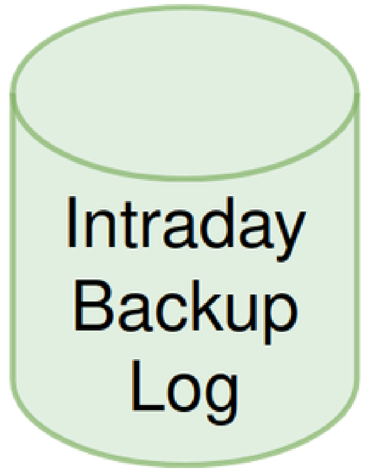


- on disk
- **backup** for rdb

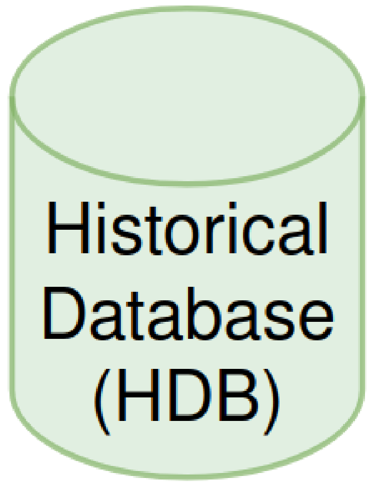
kdb+tick architecture



- in-memory
- today's data **only**

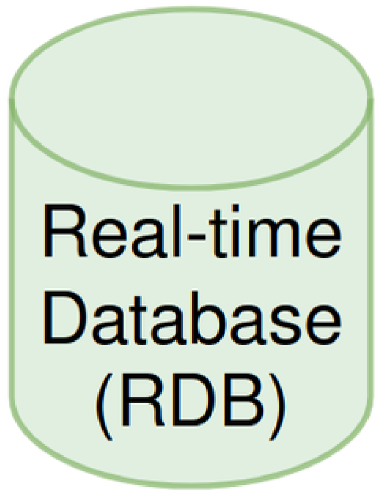


- on disk
- **backup** for rdb

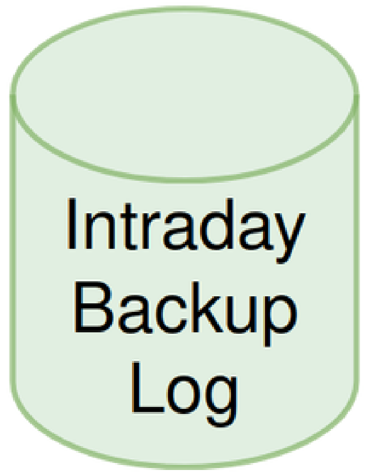


- on disk
- all historical data

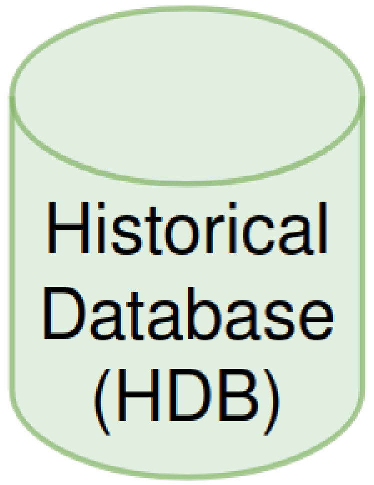
kdb+tick architecture



- in-memory
- today's data **only**



- on disk
- **backup** for rdb



- on disk
- all historical data

- main process
- takes in data
- publishes data to RDB

kdb+tick architecture

Real-time Database (RDB)

- in-memory
- today's data **only**

Intraday Backup Log

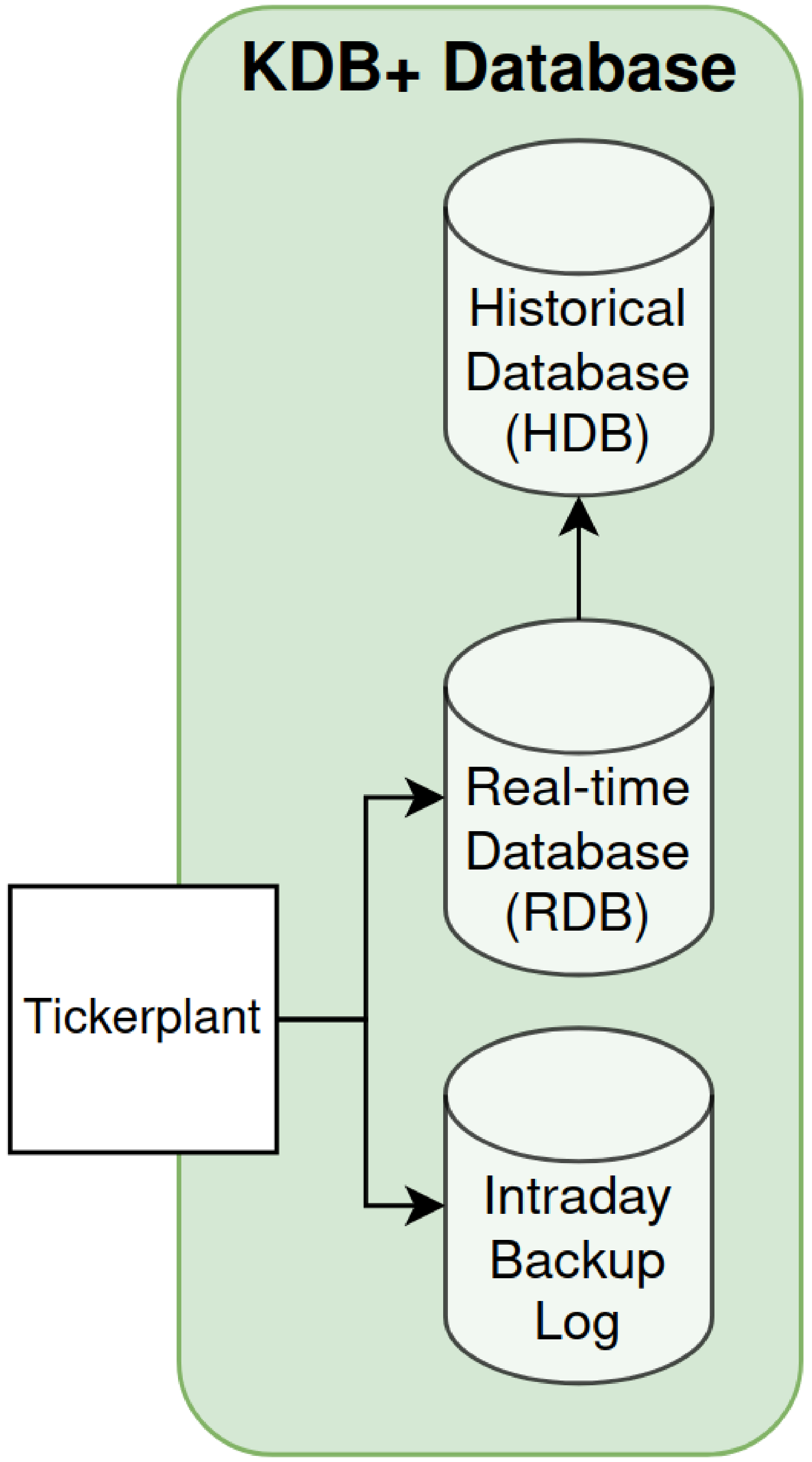
- on disk
- **backup** for rdb

Historical Database (HDB)

- on disk
- all historical data

Tickerplant

- main process
- takes in data
- publishes data to RDB



Data Feed: *Cryptofeed*

- Need **raw** exchange market data
- Need to connect through **lots of APIs**

Data Feed: *Cryptofeed*

- Need **raw** exchange market data
- Need to connect through **lots of APIs**
- Cryptofeed library creates WebSocket connections

Data Feed: *Cryptofeed*

- Need **raw** exchange market data
- Need to connect through **lots of APIs**

- Cryptofeed library creates WebSocket connections
- Formats data uniformly

Data Feed: *Cryptofeed*

- Need **raw** exchange market data
- Need to connect through **lots of APIs**

- Cryptofeed library creates WebSocket connections
- Formats data uniformly
- Supports a large number of Cryptocurrency exchanges, Spot tickers and Futures Tickers

Data Feed: *Cryptofeed*

- Need **raw** exchange market data
- Need to connect through **lots of APIs**

- Cryptofeed library creates WebSocket connections
- Formats data uniformly
- Supports a large number of Cryptocurrency exchanges, Spot tickers and Futures Tickers
- Variety of data channels providing L1, L2 Order book, Open interest data and more

Frontend

- Developed using React, TypeScript and Bootstrap

- Utilised LightningChartJS as charting library, which allowed for:
    - **Real-time** visualisations of millions of data points

Websockets & HTTP

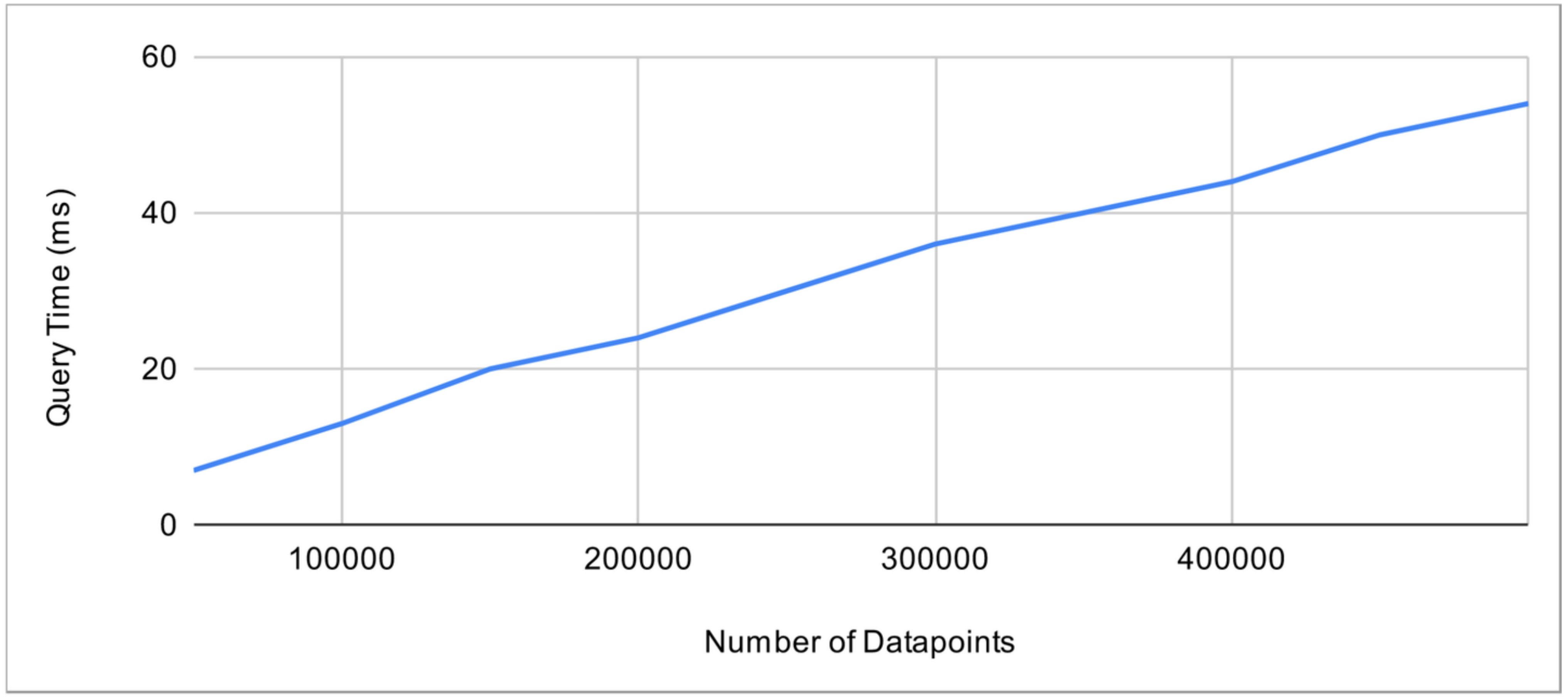
- Requirement to display real-time, high frequency data on frontend
- Websockets allow for low latency communication between frontend and server
- HTTP requests used for short term static data

Django

- Chosen to facilitate Python
- Used qPython for IPC with the q language
- Django doesn't support websockets natively, so we used an extension called Django Channels
- Redis' Message Passing is used to communicate data from Cryptofeed to Django Consumers

Evaluation

- RDB Benchmark
  - Queried 500,000 data-points which is half a days of data
  -
- Code Evaluation
  - Code reviews
  - Automated tests



kdb+ Query Benchmark

Evaluation

- Qualitative Evaluation
    - User testing
    - Alternative Perspectives

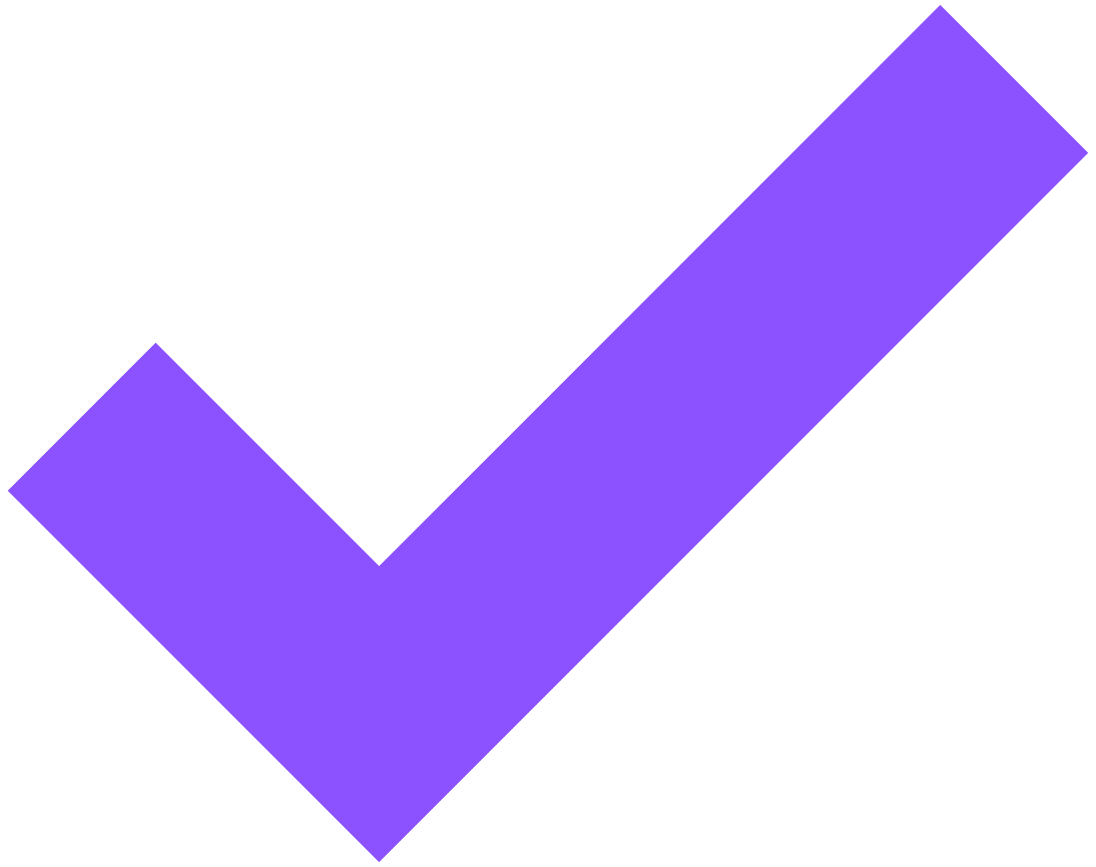
Ethical Concerns

Financial Warning

Legality of derivatives

Usage of currencies

Conclusion



Does QSync achieve the goals set out earlier in the presentation?

Conclusion

“The team has implemented a sophisticated modular system that deals with level-2 data arriving in real-time. The application provides cryptocurrency traders with a flexible general-purpose tool for summarising the cryptocurrency markets and detecting arbitrage opportunities.”

Dr. Paul Bilokon

Future Improvements

Future improvements to QSync include:

- Accessibility improvements
- More visualisations
- *Support more currencies by collecting more data*

Thank You