Stock Data Processing in Delta Lake

CSCI E-63 Big Data Analytics Spring 2021

Danil Astakhov

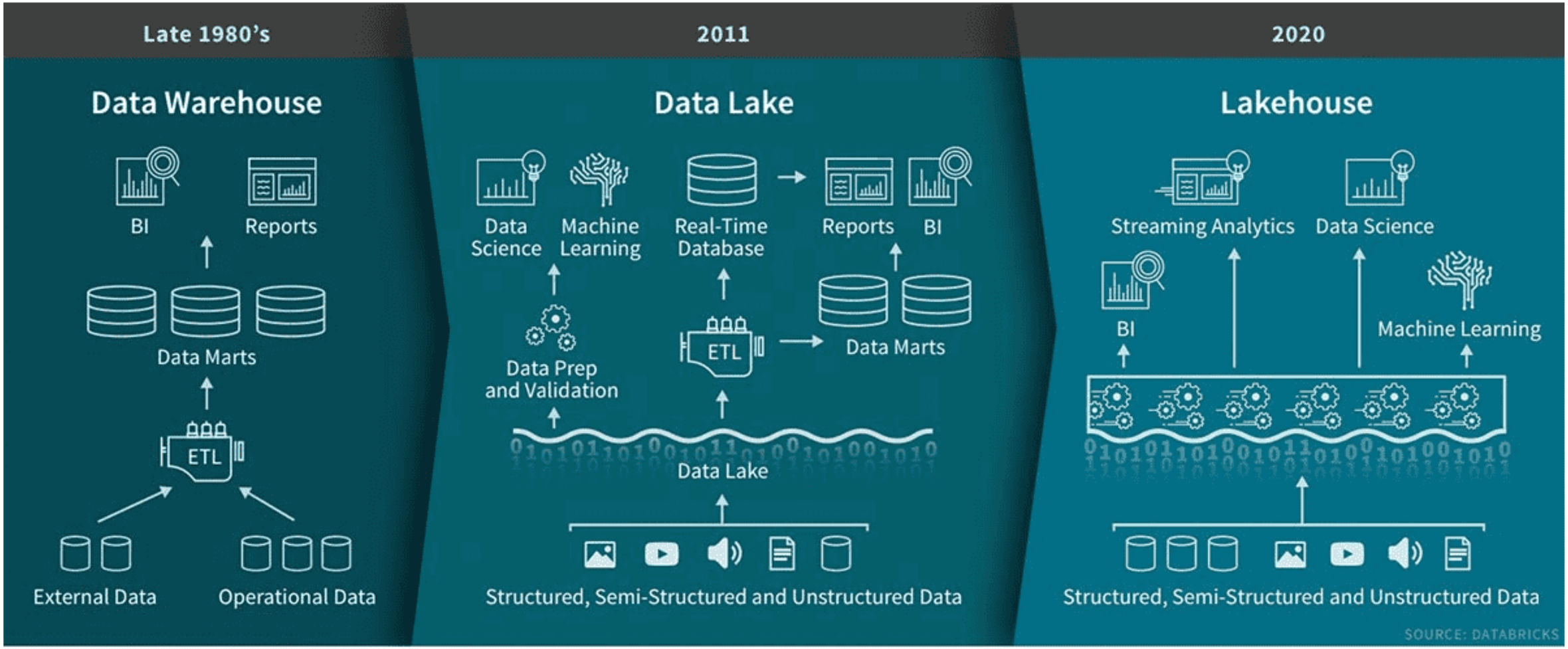
Problem statement

Architecture construction of an ACID-compliant storage and streaming processing of stock market data based on Databricks Delta Lake.

Problem description

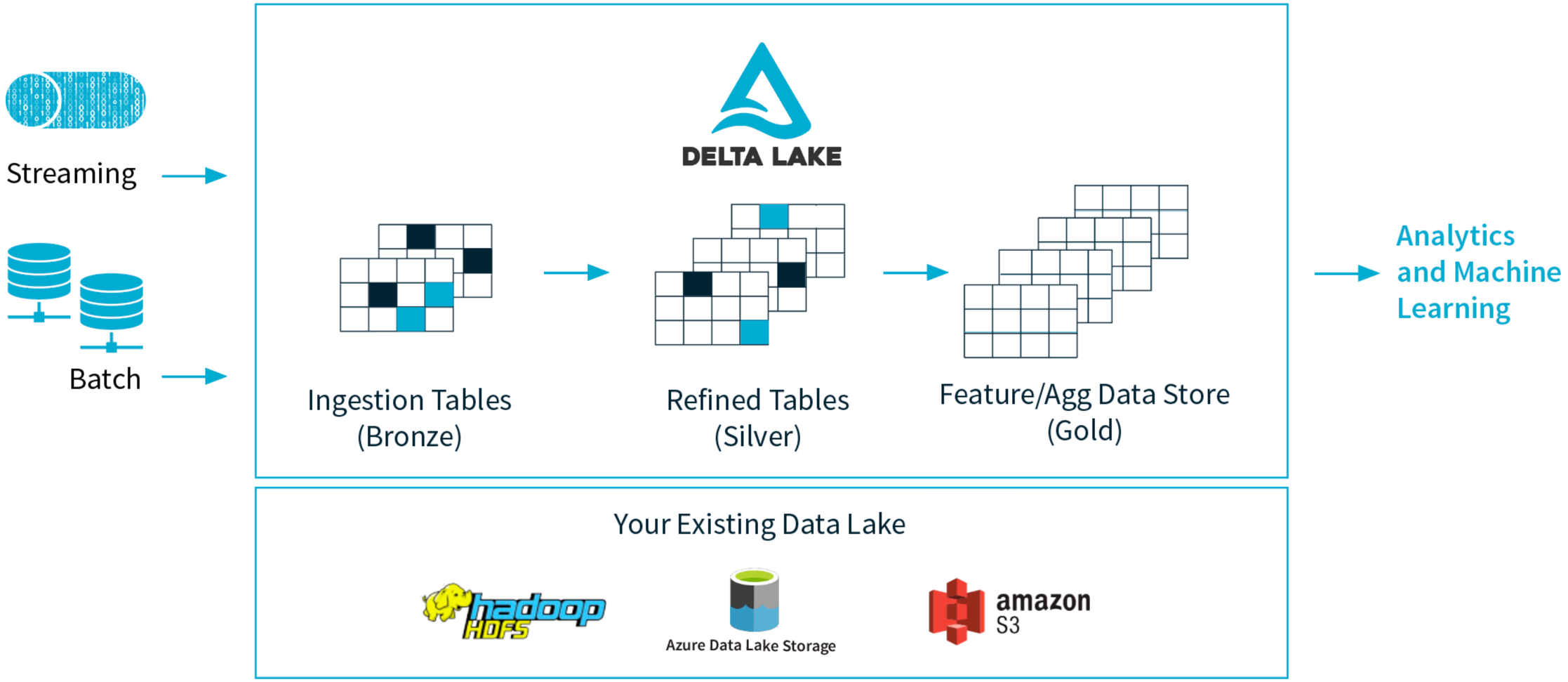
Designing the storage architecture is an essential part in big data solutions engineering. The central issue of these tasks is the choice of persistent storage. Even though Data Lake and Data Warehouse are powerful tools, the classic solutions based on them have a high complexity of application architecture. Just as difficult is the implementation of CRUD scripts in storage. Architectures based on Delta Lake are designed to eliminate these drawbacks. This paper presents the implementation of Delta Lake architecture for stock market data. The stock market is a suitable data model for data streaming, moreover, there is a wide scope for Business Intelligence.

Lakehouse architecture

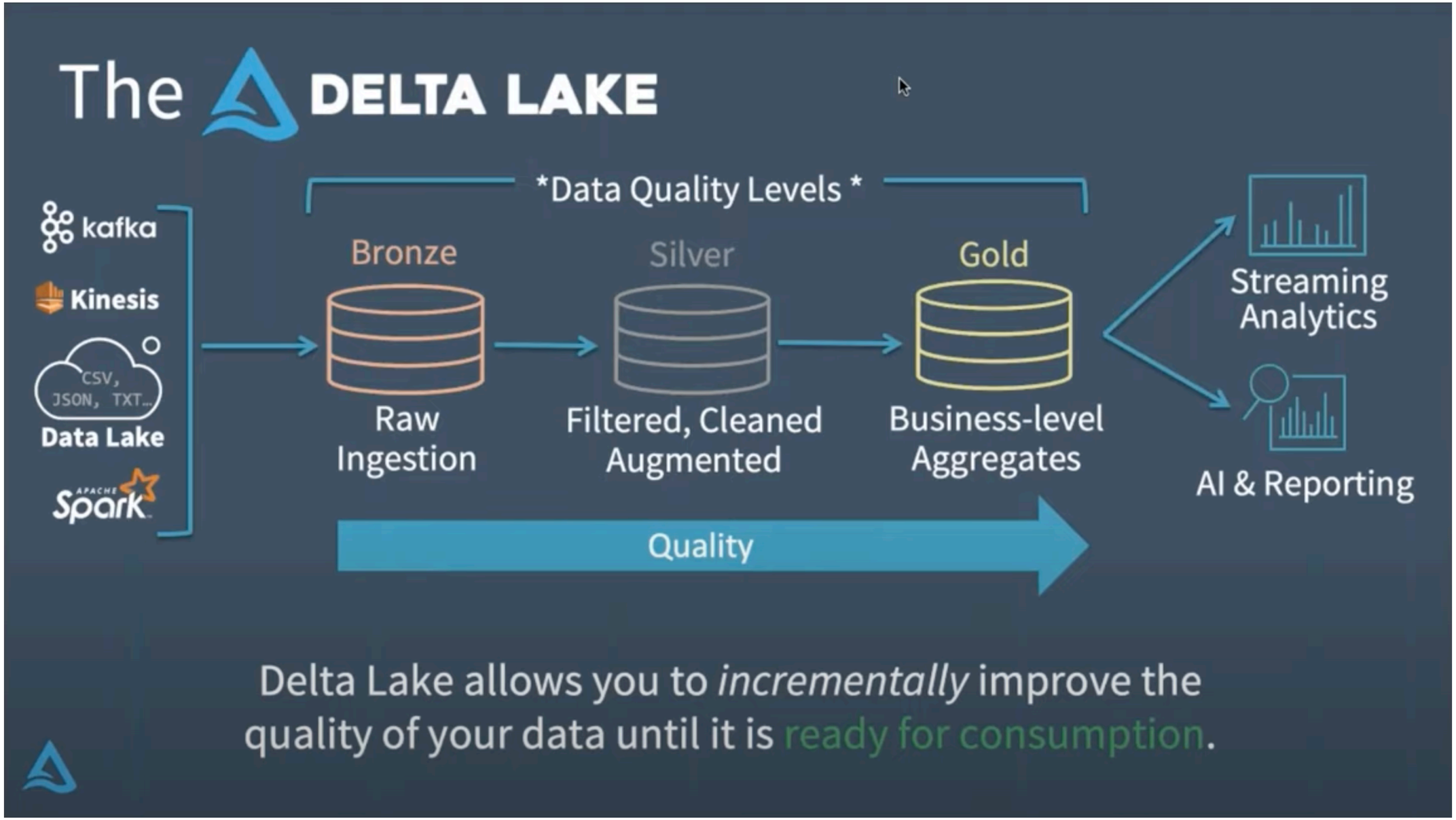


picture source: Databricks: what is a data lakehouse

Delta lake architecture



picture source: https://delta.io



picture source: Making Apache Spark™ Better with Delta Lake

Data | Tasks (1) | Code (105) | Discussion (7) | Activity | Metadata

Download (56 MB) | New Notebook

Usability 7.6 | **License** CC0: Public Domain | **Tags** business, finance, investing

Description

Context

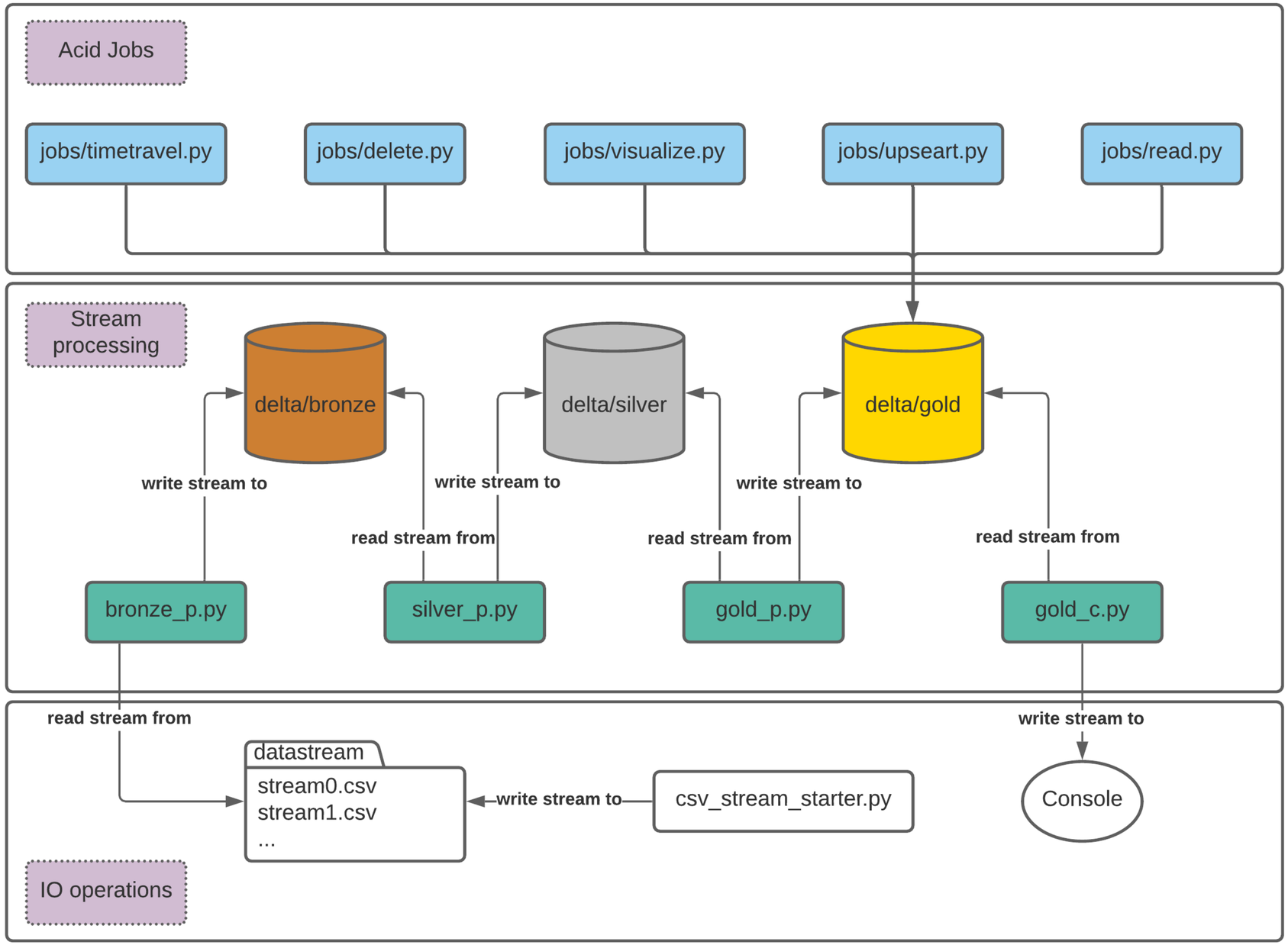
Stock market data can be interesting to analyze and as a further incentive, strong predictive models can have large financial payoff. The amount of financial data on the web is seemingly endless. A large and well structured dataset on a wide array of companies can be hard to come by. Here I provide a dataset with historical stock prices (last 5 years) for all companies currently found on the S&P 500 index.

The script I used to acquire all of these .csv files can be found in this GitHub repository
In the future if you wish for a more up to date dataset, this can be used to acquire new versions of the .csv files.

Feb 2018 note: I have just updated the dataset to include data up to Feb 2018. I have also accounted for changes in the stocks on the S&P 500 index (RIP whole foods etc. etc.).

https://www.kaggle.com/camnugent/sandp500

Solution architecture



Structured Streaming

```
silver = spark.readStream.format("delta").load("delta/silver/")

query = silver.withColumn('timestamp',
                          F.unix_timestamp(F.col('date'), "yyyy-MM-dd").cast(stypes.TimestampType())) \
    .withWatermark("timestamp", "1 minutes") \
    .select('*').groupby(silver.name, "timestamp").agg(
        F.max(F.col('high') - F.col('low')).alias('max_range'),
        F.sum(silver.volume).alias('volume_sum'),
        F.sum((F.col('open') - F.col('close')) / F.col('open') * 100).alias('delta_total_percents')
)

query = query.writeStream.format("delta").outputMode("append") \
    .option("checkpointLocation", "checkpoints/etl-third-to-fourth") \
    .option("mergeSchema", "true") \
    .start("delta/gold/")

query.awaitTermination()
```

ACID upsert

```
upseart_data = spark.read.format("csv").option("header", "true").load("../datasource/upseart.csv")

gold.alias("gold").merge(
    upseart_data.alias("upseart"),
    "gold.name = upseart.name") \
  .whenMatchedUpdate(set={"delta_total_percents": "upseart.delta_total_percents"}) \
  .whenNotMatchedInsert(values={
        "name": "upseart.name",
        "timestamp": "upseart.timestamp",
        "max_range": "upseart.max_range",
        "volume_sum": "upseart.volume_sum",
        "delta_total_percents": "upseart.delta_total_percents",
    }
  ) \
  .execute()

gold.toDF().select('*').where(F.col("delta_total_percents") > 2).show()
```

Metadata is Big Data

version	timestamp	userId	userName	operation	operationParameters	job	notebook	clusterId	readVersion	isolationLevel	isBlindAppend
74	2021-05-13 04:37:56	null	null	MERGE	[predicate -> (go...	null	null	null	73	null	false
73	2021-05-13 04:35:10	null	null	DELETE	[predicate -> ["(...	null	null	null	72	null	false
72	2021-05-13 04:20:16	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	71	null	true
71	2021-05-13 04:20:10	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	70	null	true
70	2021-05-13 04:20:02	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	69	null	true
69	2021-05-13 04:19:54	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	68	null	true
68	2021-05-13 04:19:46	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	67	null	true
67	2021-05-13 04:19:39	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	66	null	true
66	2021-05-13 04:19:33	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	65	null	true
65	2021-05-13 04:19:26	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	64	null	true
64	2021-05-13 04:19:19	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	63	null	true
63	2021-05-13 04:19:13	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	62	null	true
62	2021-05-13 04:19:05	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	61	null	true
61	2021-05-13 04:18:59	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	60	null	true
60	2021-05-13 04:18:52	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	59	null	true
59	2021-05-13 04:18:44	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	58	null	true
58	2021-05-13 04:18:35	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	57	null	true
57	2021-05-13 04:18:28	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	56	null	true
56	2021-05-13 04:18:21	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	55	null	true
55	2021-05-13 04:18:14	null	null	STREAMING UPDATE	[outputMode -> Ap...	null	null	null	54	null	true

only showing top 20 rows

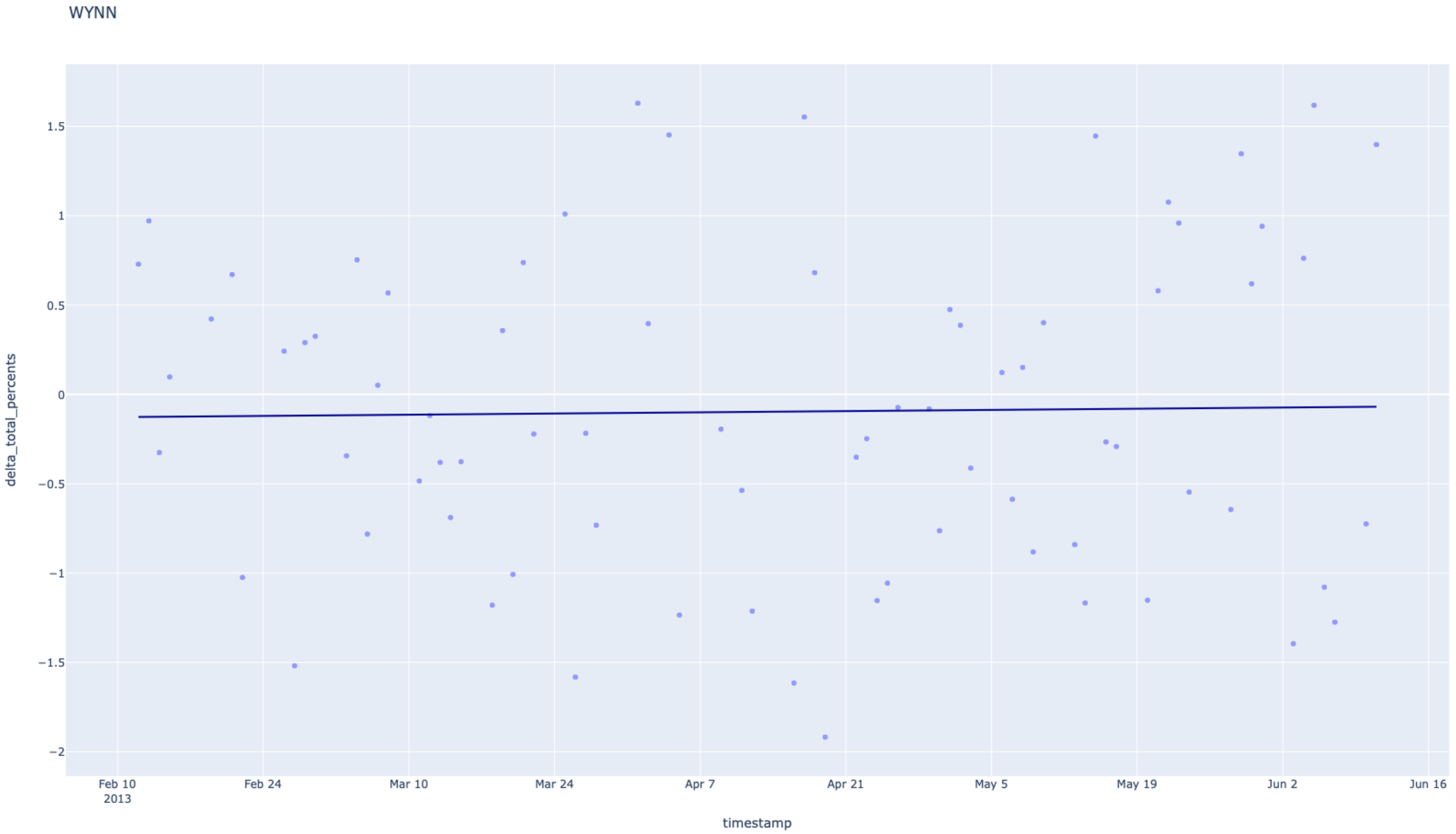
Output stream from gold layer

```
-------------------------------------------
Batch: 59
-------------------------------------------
+-----+-------------------+-------------------+-----------+--------------------+
|name |timestamp          |max_range          |volume_sum |delta_total_percents|
+-----+-------------------+-------------------+-----------+--------------------+
|CSX  |2013-06-10 00:00:00|0.279998779296875  |6823939.0  |0.31796472254066116 |
|GILD |2013-06-10 00:00:00|1.4500007629394531 |8744157.0  |2.0728303009635902  |
|STZ  |2013-06-10 00:00:00|0.9500007629394531 |1377841.0  |1.2715039472630223  |
|BLL  |2013-06-10 00:00:00|0.1849994659423828 |908044.0   |0.5040119427548982  |
|DOV  |2013-06-10 00:00:00|1.28900146484375   |920249.0   |1.1665834055886366  |
|ESS  |2013-06-10 00:00:00|2.9399871826171875 |298738.0   |0.08326707150882279 |
|PM   |2013-06-10 00:00:00|1.3199996948242188 |3888447.0  |0.6828580839205263  |
|ROP  |2013-06-10 00:00:00|1.0                |301264.0   |0.37817929298324066 |
|APC  |2013-06-10 00:00:00|1.0999984741210938 |1886003.0  |0.04539368393164887 |
|YUM  |2013-06-10 00:00:00|1.69000244140625   |3975062.0  |1.2670303621831502  |
|KMX  |2013-06-10 00:00:00|0.7462005615234375 |1458434.0  |0.2330521368082962  |
|AET  |2013-06-10 00:00:00|0.8987998962402344 |2197540.0  |-1.1446623347365277 |
|ANSS |2013-06-10 00:00:00|1.410003662109375  |259039.0   |1.1772538507263797  |
|PFE  |2013-06-10 00:00:00|0.31999969482421875|6.6492012E7|0.03523016853018487 |
|WY   |2013-06-10 00:00:00|0.7399997711181641 |5194843.0  |1.3559309102721133  |
|VMC  |2013-06-10 00:00:00|1.9799995422363281 |501905.0   |-2.022896045280164  |
|DISCA|2013-06-10 00:00:00|2.2099990844726562 |1497131.0  |2.5940040736747814  |
|XLNX |2013-06-10 00:00:00|0.6399993896484375 |3002325.0  |0.7132339316666315  |
|ETR  |2013-06-10 00:00:00|1.1299972534179688 |1607675.0  |-0.40409694847345495|
|IDXX |2013-06-10 00:00:00|0.6899986267089844 |744852.0   |0.17828824602544624 |
+-----+-------------------+-------------------+-----------+--------------------+
```

```
-------------------------------------------
Batch: 60
-------------------------------------------
+----+-------------------+-------------------+----------+--------------------+
|name|timestamp          |max_range          |volume_sum|delta_total_percents |
+----+-------------------+-------------------+----------+--------------------+
|INTC|2013-06-11 00:00:00|0.4500007629394531 |2.70765E7 |0.48328972856270336  |
|MET |2013-06-11 00:00:00|0.8100013732910156 |7923089.0 |0.9133434016188798   |
|NBL |2013-06-11 00:00:00|1.1300010681152344 |1528841.0 |-0.08663877638840507 |
|GS  |2013-06-11 00:00:00|3.2079925537109375 |3858073.0 |1.2337517203626525   |
|AMGN|2013-06-11 00:00:00|2.3886032104492188 |2064473.0 |-0.6887706035012656  |
|PCLN|2013-06-11 00:00:00|12.3399658203125   |551130.0  |0.23304032735540853  |
|SBUX|2013-06-11 00:00:00|0.3975028991699219 |7897496.0 |0.07637551676415791  |
|BAC |2013-06-11 00:00:00|0.21000003814697266|1.06242E8 |-0.07627939850282585 |
|CERN|2013-06-11 00:00:00|0.9520034790039062 |995212.0  |-0.3681740794087492  |
|PRGO|2013-06-11 00:00:00|1.970001220703125  |408550.0  |-0.6529788987163293  |
|MGM |2013-06-11 00:00:00|0.23999977111816406|5931244.0 |0.8713144354970362   |
|INTU|2013-06-11 00:00:00|1.0                |2427498.0 |0.9946867045013623   |
|SNA |2013-06-11 00:00:00|1.5099945068359375 |249324.0  |0.5286962794286098   |
|UAL |2013-06-11 00:00:00|0.9200000762939453 |4974048.0 |-0.9331235546569561  |
|KORS|2013-06-11 00:00:00|1.5289993286132812 |2576058.0 |0.9377494432762211   |
|NTAP|2013-06-11 00:00:00|0.9099998474121094 |6162031.0 |-1.3004189485163682  |
|TSCO|2013-06-11 00:00:00|1.3499984741210938 |927700.0  |-1.7166239437312645  |
|CBS |2013-06-11 00:00:00|0.779998779296875  |5977822.0 |0.6120745229837115   |
|ALL |2013-06-11 00:00:00|0.93499755859375   |4054032.0 |-0.6175488100110136  |
|ANTM|2013-06-11 00:00:00|1.6399993896484375 |1814434.0 |-0.012752974972140592|
+----+-------------------+-------------------+----------+--------------------+
```

Gold layer writing mode: "append"

Gold level visualisation



Summary

Databricks DeltaLake is a new chapter in the development of Big Data solutions.

Combination of

- ACID-compatibility
- storage organisation transparency
- batch-stream processing
- convenient DataFrame interface
- metadata as DataFrame

opens up opportunities for building extremely large, scalable and durable storages and Big Data processing systems.

YouTube Brief Demo https://youtu.be/DfgN8cHtpLg

YouTube Full Demo https://youtu.be/4nye3nRV1R4

Github: https://github.com/vo0xr0c/deltalake