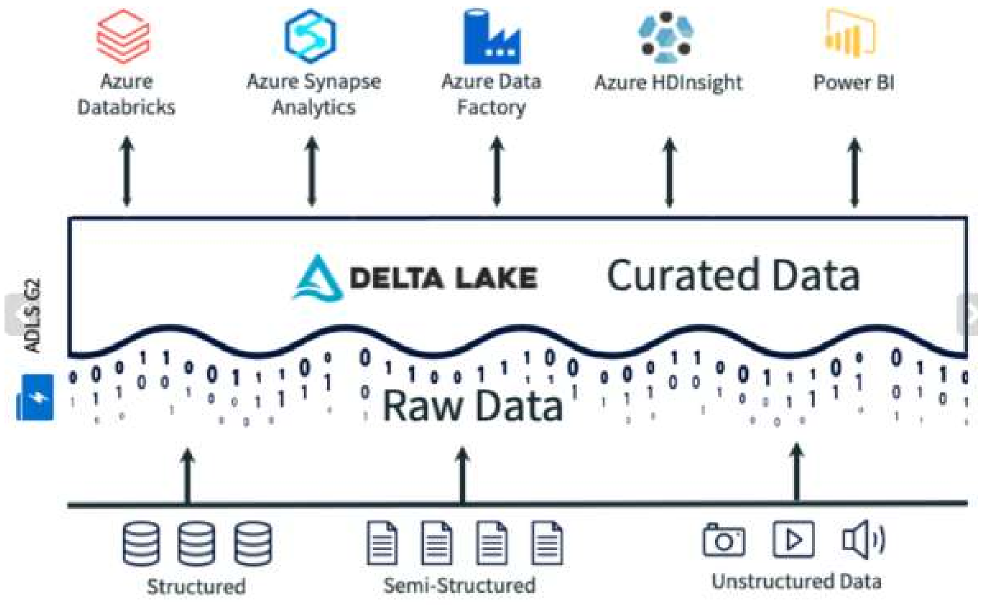
Deltalake

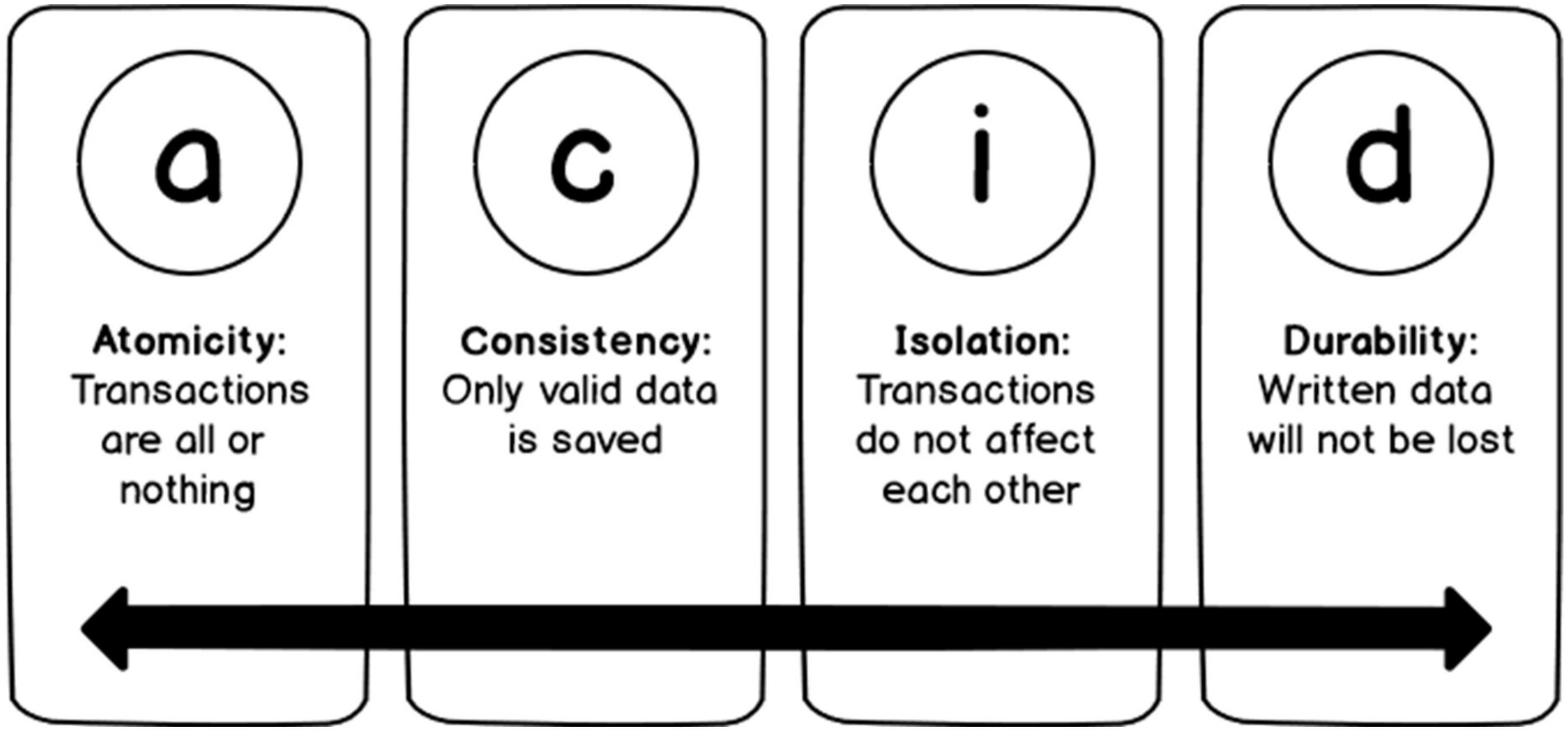
Delta Lake

- An open-source storage layer on top of data lake
- Brings ACID transaction capabilities
- Enables Delta Lake to overcome challenges in terms of delete, upserts, merge, etc
- Once the data in the data lake is stored in Delta Format it can be accessed by a variety of AWS services.

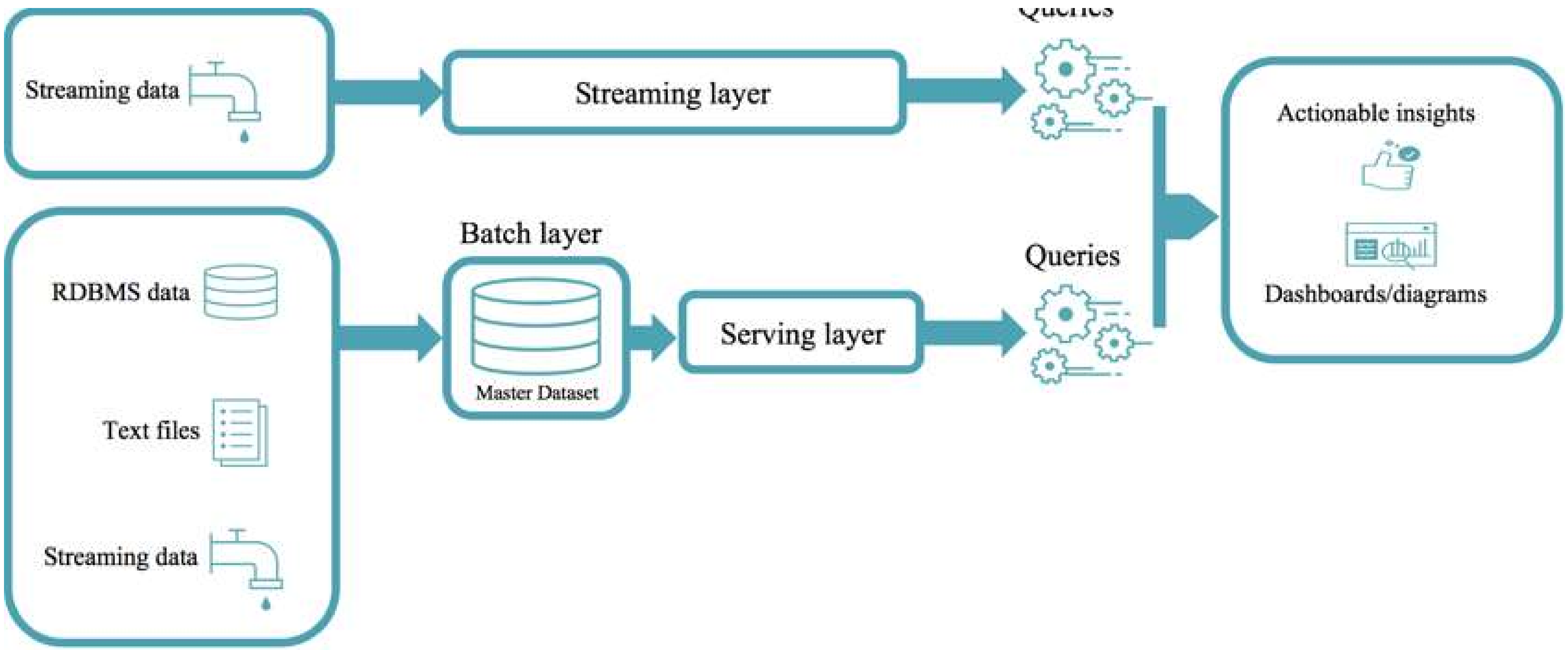


Delta Lake

- Its core functionalities bring reliability to the big data lakes by ensuring data integrity with ACID transactions


a
c
i
d
Atomicity: Transactions are all or nothing
Consistency: Only valid data is saved
Isolation: Transactions do not affect each other
Durability: Written data will not be lost

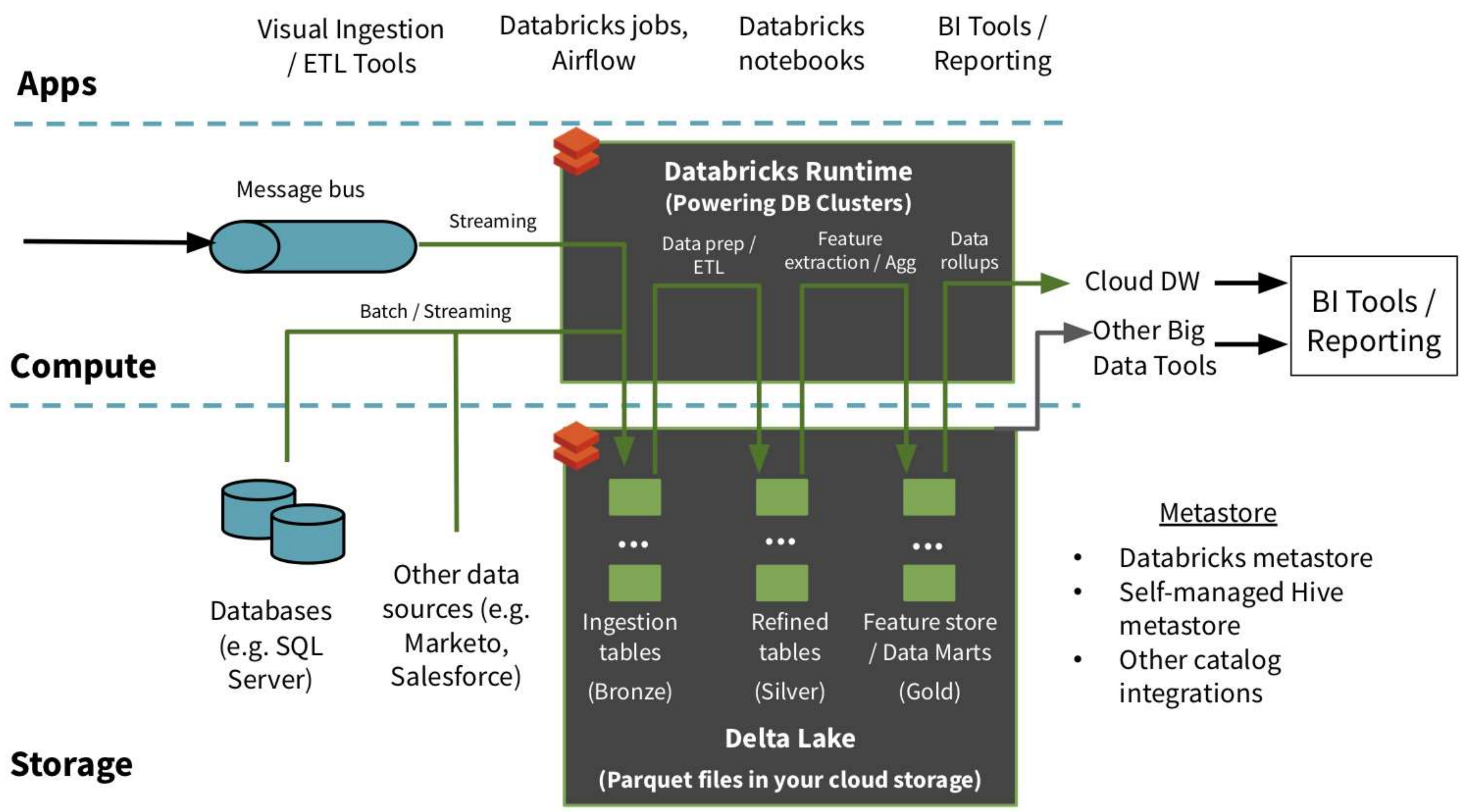
Lambda architecture



Delta Lake architecture

- Vast improvement upon the traditional Lambda architecture
- At each stage, we enrich our data
- Allows us to combine batch and streaming workflows
- **Bronze tables** contain raw data ingested from various sources
- **Silver tables** will provide a more refined view of our data
  - Can join fields from various bronze tables to enrich streaming records, or
  - Update account statuses based on recent activity.
- **Gold tables** provide business level aggregates often used for reporting and dashboarding
  - This would include aggregations such as daily active website users, weekly sales per store, or gross revenue per quarter by department.
  - The end outputs are actionable insights, dashboards, and reports of business metrics.

Delta Lake architecture



<u>Metastore</u>

- Databricks metastore
- Self-managed Hive metastore
- Other catalog integrations

Table batch reads and writes

- Supports most of the options provided by Apache Spark DataFrame read and write APIs for performing batch reads and writes on tables.

Create a table

- Delta Lake supports creating two types of tables
  - Tables defined in the metastore and
  - Tables defined by path
- You can create tables in the following ways.
  - SQL DDL commands

```
CREATE IF NOT EXISTS TABLE events (
  date DATE,
  eventId STRING,
  eventType STRING,
  data STRING)
USING DELTA
```

```
CREATE OR REPLACE TABLE events (
  date DATE,
  eventId STRING,
  eventType STRING,
  data STRING)
USING DELTA
```

Create a table

- In Databricks Runtime 7.0 and above, SQL also supports a creating table at a path without creating an entry in the Hive metastore
  - -- Create or replace table with path
  - CREATE OR REPLACE TABLE delta.`/mnt/delta/events` (
  -   date DATE,
  -   eventId STRING,
  -   eventType STRING,
  -   data STRING)
  - USING DELTA

DataFrameWriter API

- To simultaneously create a table and insert data into it, can use the Spark DataFrameWriter
  - # Create table in the metastore using DataFrame's schema and write data to it
  - df.write.format("delta").saveAsTable("events")

  - # Create or replace partitioned table with path using DataFrame's schema and write/overwrite data to it
  - df.write.format("delta").mode("overwrite").save("/mnt/delta/events")

Control data location

- For tables defined in the metastore, you can optionally specify the LOCATION as a path
- Tables created with a specified LOCATION are considered unmanaged by the metastore
- Unlike a managed table, where no path is specified, an unmanaged table's files are not deleted when you DROP the table
    - CREATE TABLE events
    - USING DELTA
    - LOCATION '/mnt/delta/events'

Thanks