References:

<https://www.bing.com/search?q=data+lake+vs+delta+lake&qs=AS&pq=data+lake+vs+delta+&sc=10-19&cvid=F578336C4ED14A24BE748BF13D0B4A33&FORM=QBRE&sp=1&ghc=1>

<https://learn.microsoft.com/en-us/azure/databricks/ingestion/auto-loader/options>

<https://docs.databricks.com/workflows/delta-live-tables/delta-live-tables-concepts.html>

<https://www.youtube.com/watch?v=kNG-5uRY4o8> – This is about Autoloader

<https://www.youtube.com/watch?v=0K_CLhwRHAM> – This is about delta live tables

**Delta Lake** is an open-source storage layer that brings ACID (atomicity, consistency, isolation, and durability) transactions to Apache Spark and big data workloads.

The current version of Delta Lake included with Azure Synapse has language support for Scala, PySpark, and .NET and is compatible with Linux Foundation Delta Lake. There are links at the bottom of the page to more detailed examples and documentation.

**Delta Live Tables** (DLT) makes it easy to build and manage reliable data pipelines that deliver high-quality data on Delta Lake. DLT helps data engineering teams simplify ETL development and management with declarative pipeline development, automatic data testing, and deep visibility for monitoring and recovery.

**More easily build and maintain data pipelines:**

With Delta Live Tables, easily define end-to-end data pipelines by specifying the data source, the transformation logic and destination state of the data — instead of manually stitching together siloed data processing jobs. Automatically maintain all data dependencies across the pipeline and reuse ETL pipelines with environment independent data management. Run in batch or streaming and specify incremental or complete computation for each table.

**Automatic testing:**

Delta Live Tables helps to ensure accurate and useful BI, data science and machine learning with high-quality data for downstream users. Prevent bad data from flowing into tables through validation and integrity checks and avoid data quality errors with predefined error policies (fail, drop, alert or quarantine data). In addition, you can monitor data quality trends over time to get insight into how your data is evolving and where changes may be necessary.

**Deep visibility for monitoring and easy recovery:**

Gain deep visibility into pipeline operations with tools to visually track operational stats and data lineage. Reduce downtime with automatic error handling and easy replay. Speed up maintenance with single-click deployment and upgrades.

Example: CREATE STREAMING LIVE TABLE customers

COMMENT "The customers buying finished products, ingested from /databricks-datasets."

TBLPROPERTIES ("quality" = "mapping")

AS SELECT \* FROM cloud\_files("/databricks-datasets/retail-org/customers/", "csv");

CREATE STREAMING LIVE TABLE sales\_orders\_raw

COMMENT "The raw sales orders, ingested from /databricks-datasets."

TBLPROPERTIES ("quality" = "bronze")

AS

SELECT \* FROM cloud\_files("/databricks-datasets/retail-org/sales\_orders/", "json", map("cloudFiles.inferColumnTypes", "true"));

**What is Auto Loader**

Auto Loader incrementally and efficiently processes new data files as they arrive in cloud storage without any additional setup.

**How does Auto Loader work?**

Auto Loader incrementally and efficiently processes new data files as they arrive in cloud storage. Auto Loader can load data files from AWS S3 (s3://), Azure Data Lake Storage Gen2 (ADLS Gen2, abfss://), Google Cloud Storage (GCS, gs://), Azure Blob Storage (wasbs://), ADLS Gen1 (adl://), and Databricks File System (DBFS, dbfs:/). Auto Loader can ingest JSON, CSV, PARQUET, AVRO, ORC, TEXT, and BINARYFILE file formats.

Auto Loader provides a Structured Streaming source called cloudFiles. Given an input directory path on the cloud file storage, the cloudFiles source automatically processes new files as they arrive, with the option of also processing existing files in that directory.

**Benefits of Auto Loader over using Structured Streaming directly on files:**

Scalability

Performance

Cost effective

**Photon runtime:**

Photon is the native vectorized query engine on Azure Databricks, written to be directly compatible with Apache Spark APIs so it works with your existing code. It is developed in C++ to take advantage of modern hardware, and uses the latest techniques in vectorized query processing to capitalize on data- and instruction-level parallelism in CPUs, enhancing performance on real-world data and applications-—all natively on your data lake. Photon is part of a high-performance runtime that runs your existing SQL and DataFrame API calls faster and reduces your total cost per workload. Photon is used by default in Databricks SQL warehouses.

Photon advantages

* Supports SQL and equivalent DataFrame operations against Delta and Parquet tables.
* Accelerates queries that process a significant amount of data (100GB+) and include aggregations and joins.
* Faster performance when data is accessed repeatedly from the disk cache.
* More robust scan performance on tables with many columns and many small files.
* Faster Delta and Parquet writing using UPDATE, DELETE, MERGE INTO, INSERT, and CREATE TABLE AS SELECT, especially for wide tables (hundreds to thousands of columns).
* Replaces sort-merge joins with hash-joins.