Delta table versioning

databricks

Lesson goals

- Use the history and version of Delta tables to explore the commits made to the table.

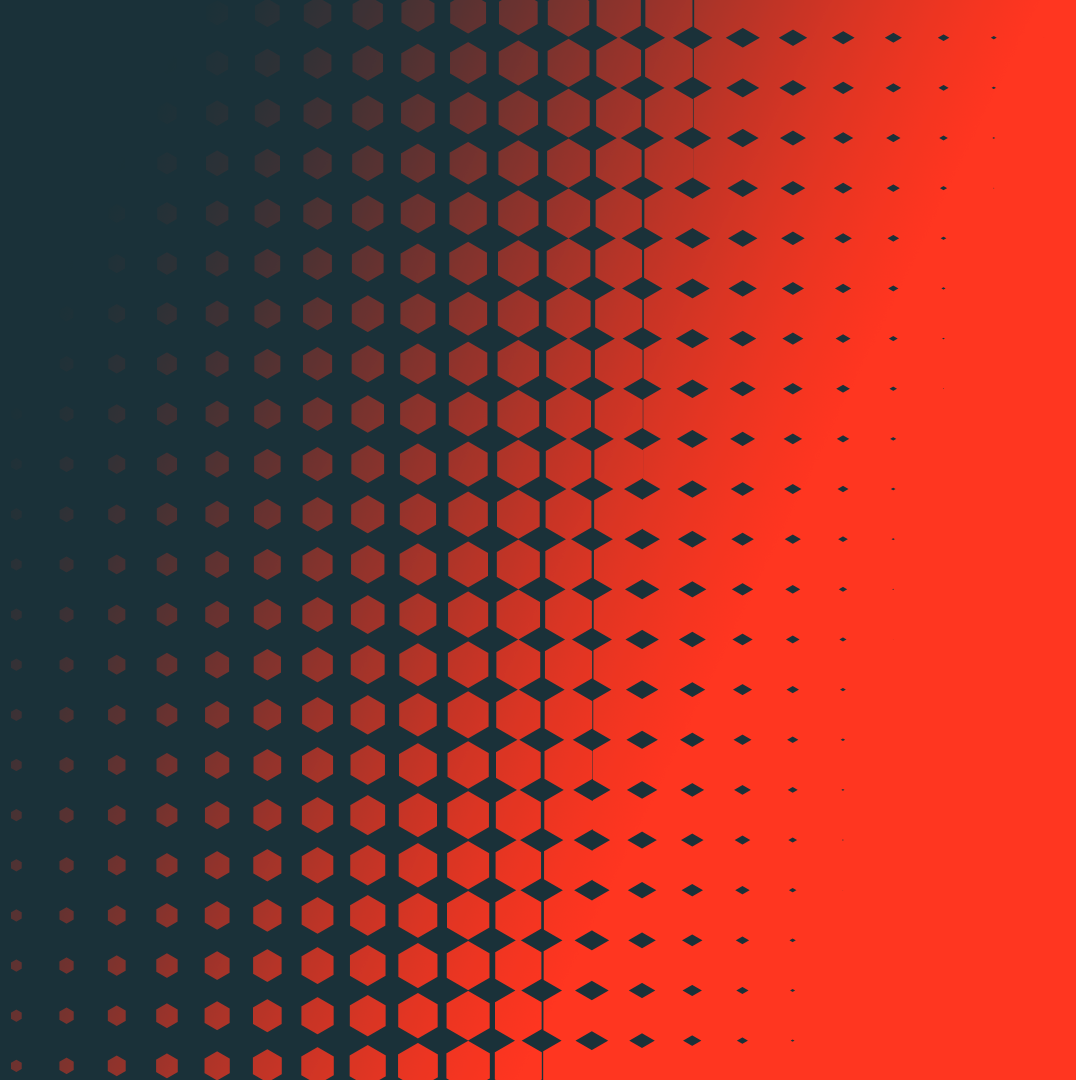
Components of a Delta Table

Parquet files in object storage

Transaction logs in object storage

Registration in a metastore (optional)

Delta transaction log



databricks

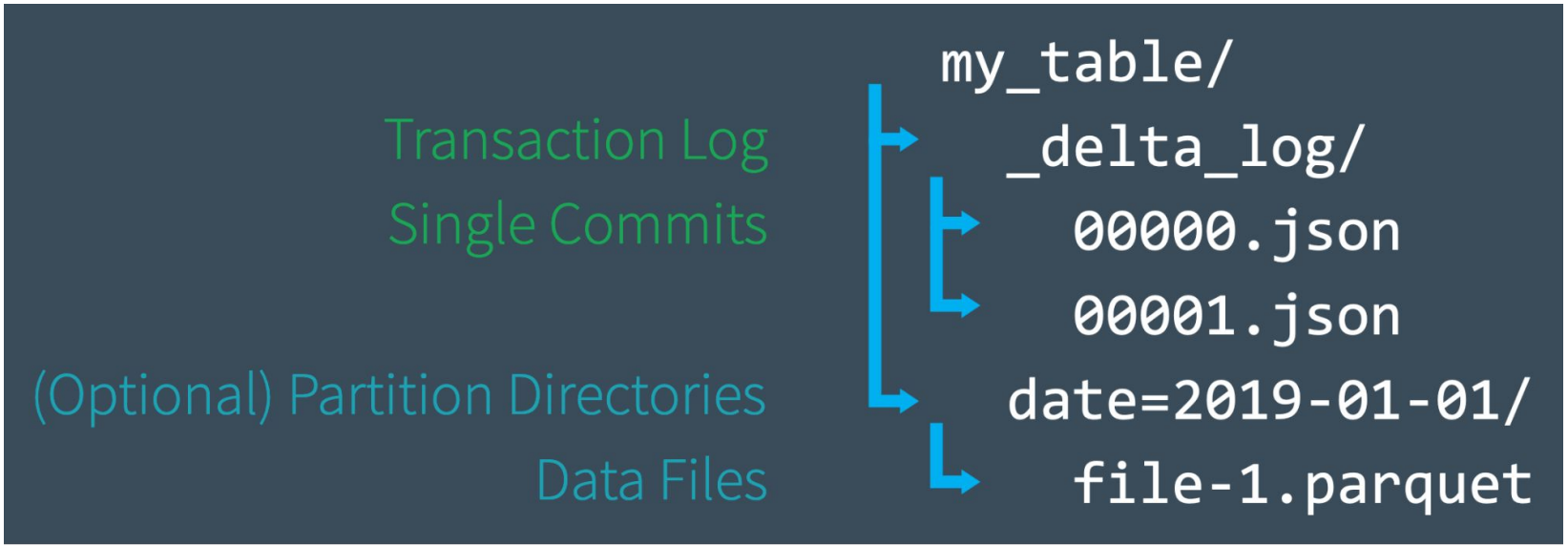
What is the Delta transaction log?

- Ordered record of the transactions performed on a Delta table
- Single source of truth for that table
- How Delta Lake guarantees atomicity

How does the transaction log work?

- Delta Lake breaks operations down into one or more of these steps:
  - Add file
  - Remove file
  - Update metadata
  - Set transaction
  - Change protocol
  - Commit info

Delta transaction log at the file level



Adding commits to the transaction log

Spark SQL `DESCRIBE HISTORY` command

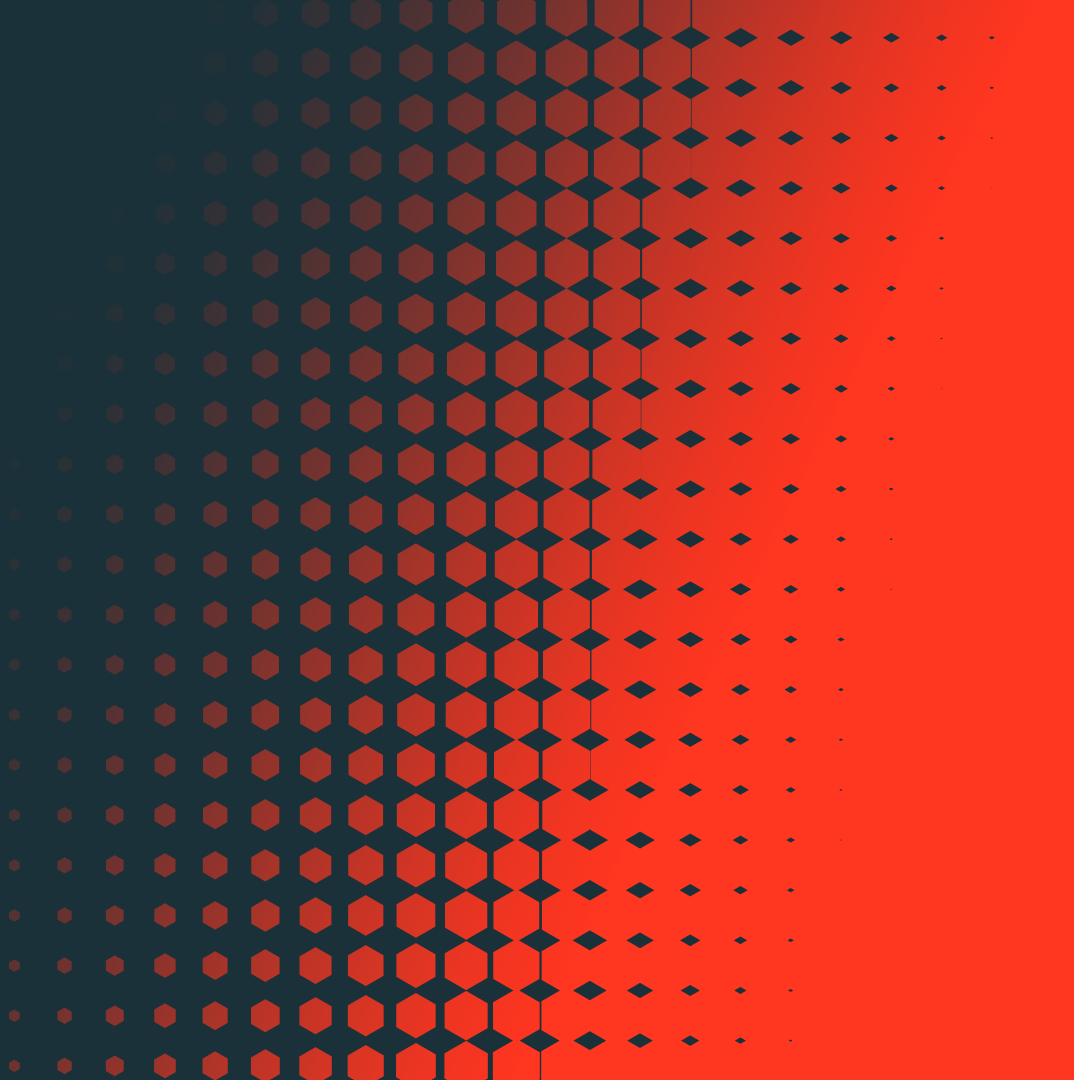
`DESCRIBE` Command

- Returns the metadata of an existing table
  - Ex. Column names, data types, comments

`DESCRIBE HISTORY` Command

- Returns a more complete set of metadata for a Delta table
  - Operation, user, operation metrics

Time Travel



databricks

Delta Lake Time Travel

- Query an older snapshot of a Delta table
  - Re-creating analysis, reports or outputs
  - Writing complex temporal queries
  - Fixing mistakes in data
  - Providing snapshot isolation

Time Travel SQL syntax

```
SELECT * FROM events TIMESTAMP AS OF timestamp_expression
SELECT * FROM events VERSION AS OF version
```

Time Travel SQL syntax

```
SELECT * FROM events TIMESTAMP AS OF timestamp_expression
```

- `timestamp_expression` can be:
  - String cast to a timestamp
  - Explicit timestamp cast
  - Date string
  - In Databricks Runtime 6.6

Time Travel SQL syntax

```
SELECT * FROM events VERSION AS OF version
```

- `Version` can be obtained from the output of `DESCRIBE HISTORY` events.

Time travel DataFrame syntax

```
df1 = (
  spark.read
  .format("delta")
  .option("timestampAsOf", timestamp_string)
  .load("/mnt/delta/events")
)
df2 = (
  spark.read
  .format("delta")
  .option("versionAsOf", version)
  .load("/mnt/delta/events")
)
```

Time Travel @ Syntax

```
# table on 2019-01-01 00:00:00.000
spark.read.format("delta").load("/mnt/delta/events@20190101000000000")
spark.read.format("delta").load("/mnt/delta/events@v123")
```


databricks