CETERIS AG

databricks

An Introduction to Azure Delta Lake using Databricks

Ceteris AG

CETERIS AG

Competences

data and AI-solutions based on Microsoft technologies
development and operation of cloud/on-prem platforms
corporate information design (IBCS)

Team

small but capable team of experienced data analytics consultants
self-organized team without typical hierarchies
agile project approach with close customer coupling

Partners

itacs GmbH
Microsoft Goldpartner
- Microsoft Power BI Partner
- Azure Marketplace und Microsoft AppSource Publisher

Certified Cubeware Partner
Certified Zebra BI Partner
GAPTEQ Partner
graphomate Partner

Speaker

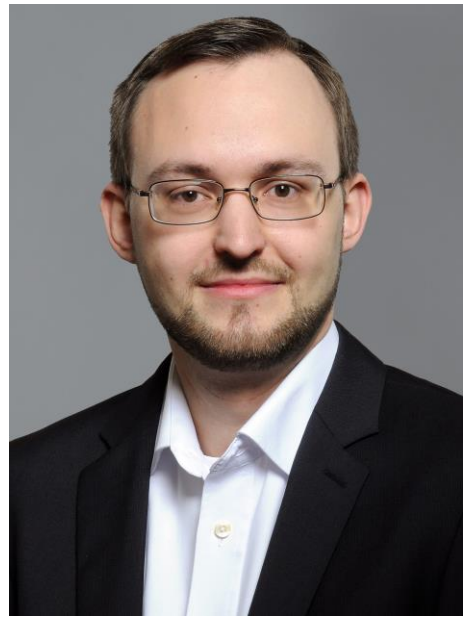
CETERIS AG

Tarek Salha

- Senior Consultant, at Ceteris AG since 2015
- Msc. Physics
- Topics:
  - Data Warehousing
  - Advanced Analytics
  - Data Lake Architectures
  - Definitely no specialist for visualization

Thorsten Huss

- Msc. Business Informatics
- Started at Ceteris AG in 2013 as student employee, now Senior Consultant
- Topics:
  - Data Integration
  - …but pretty much everything ETL, really.



itacs

Microsoft Partner
Microsoft
Gold Data Analytics
Gold Data Platform
Gold Cloud Platform

cubeware
CERTIFIED PARTNER

zebra bi

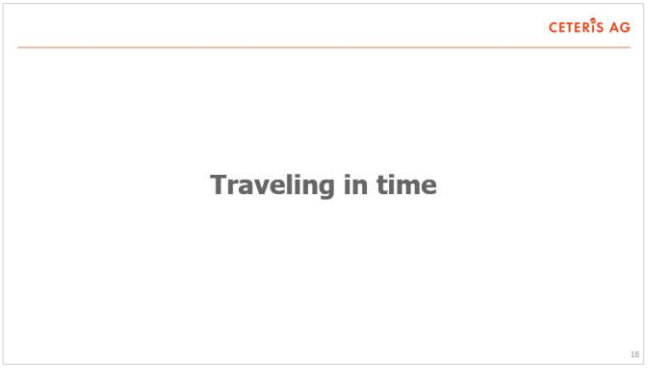
graphomate

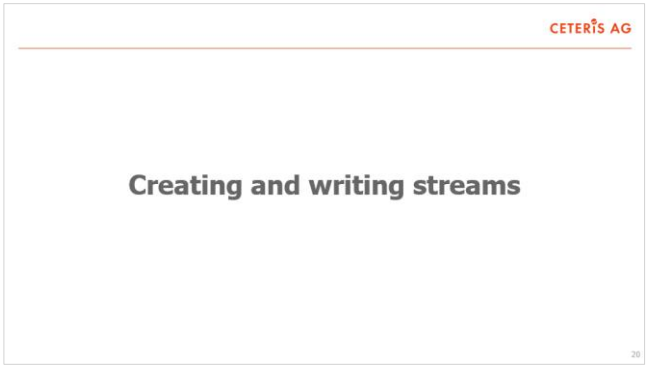
GAPTEQ

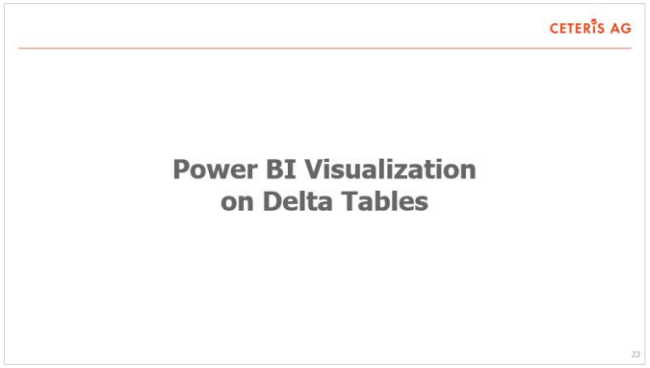
ibb HICHERT+FAISST

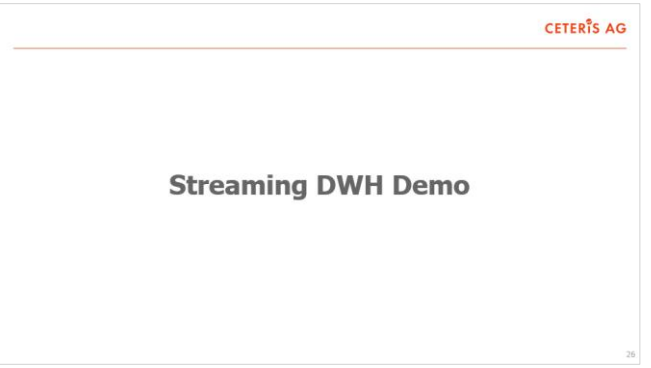
Agenda

CETERIS AG







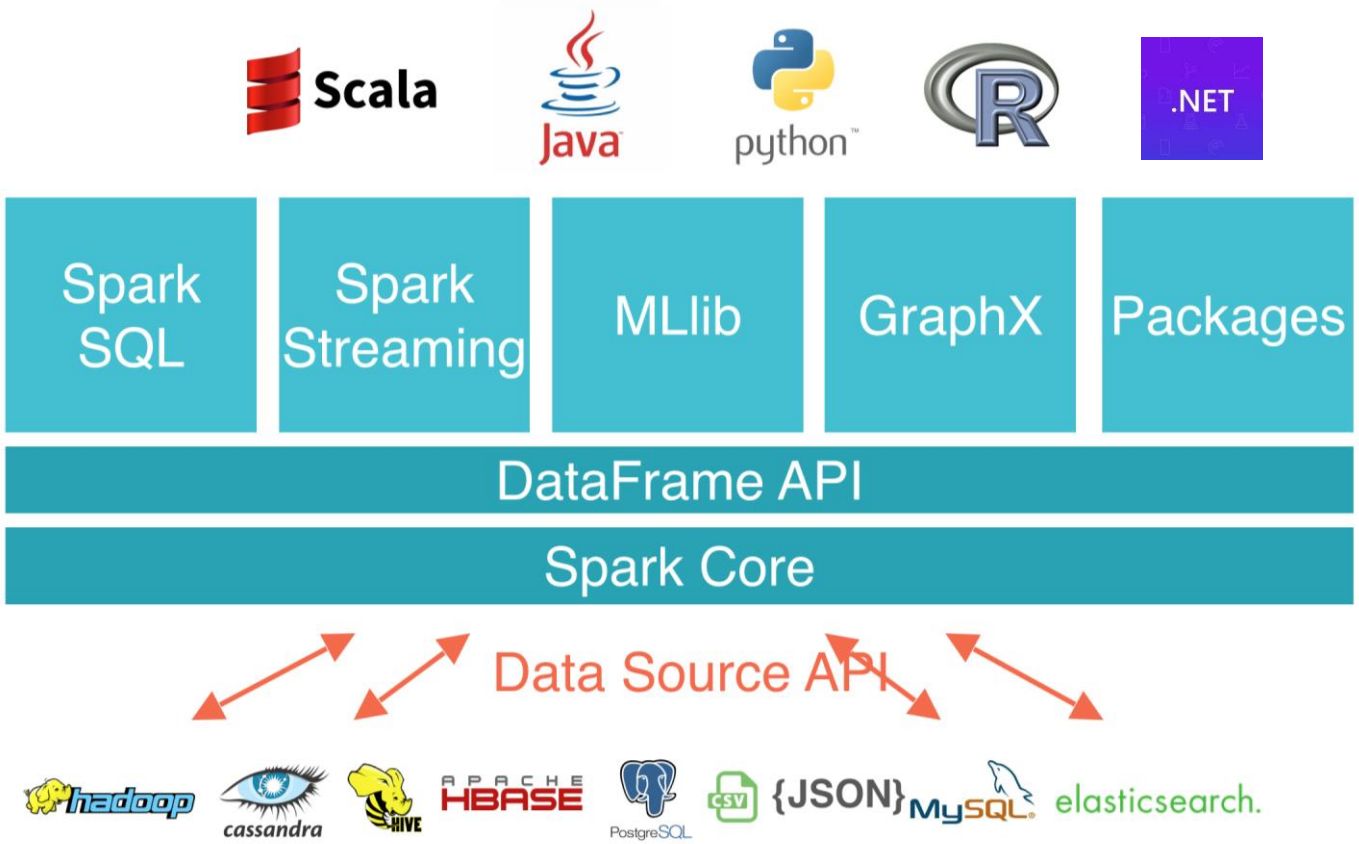


What is …?

What is Apache Spark?

CETERIS AG

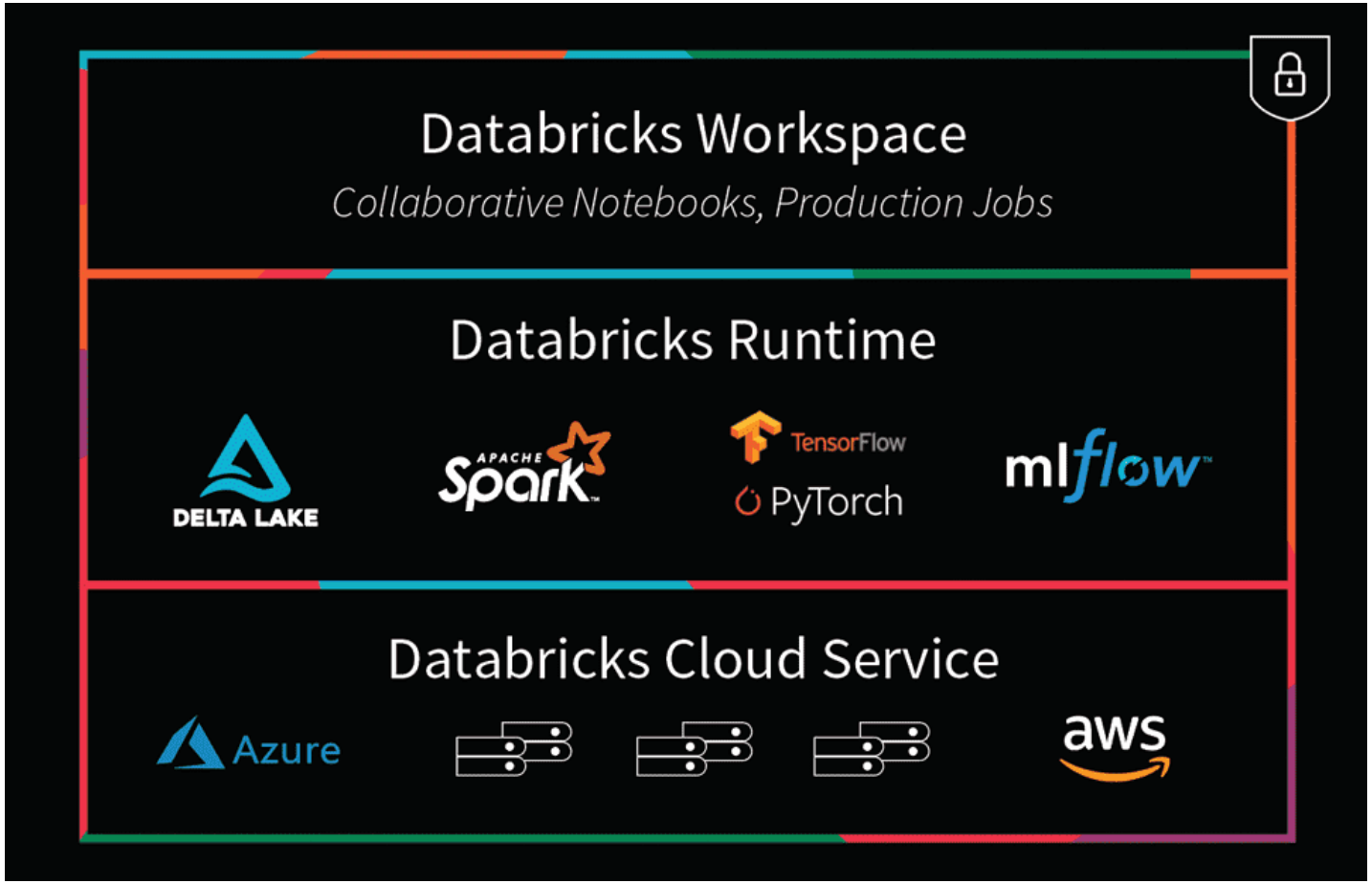
- Apache Spark is an analytics software framework, that combines cluster data processing and AI
- One of the most actively developed open source big data projects



What is Databricks?

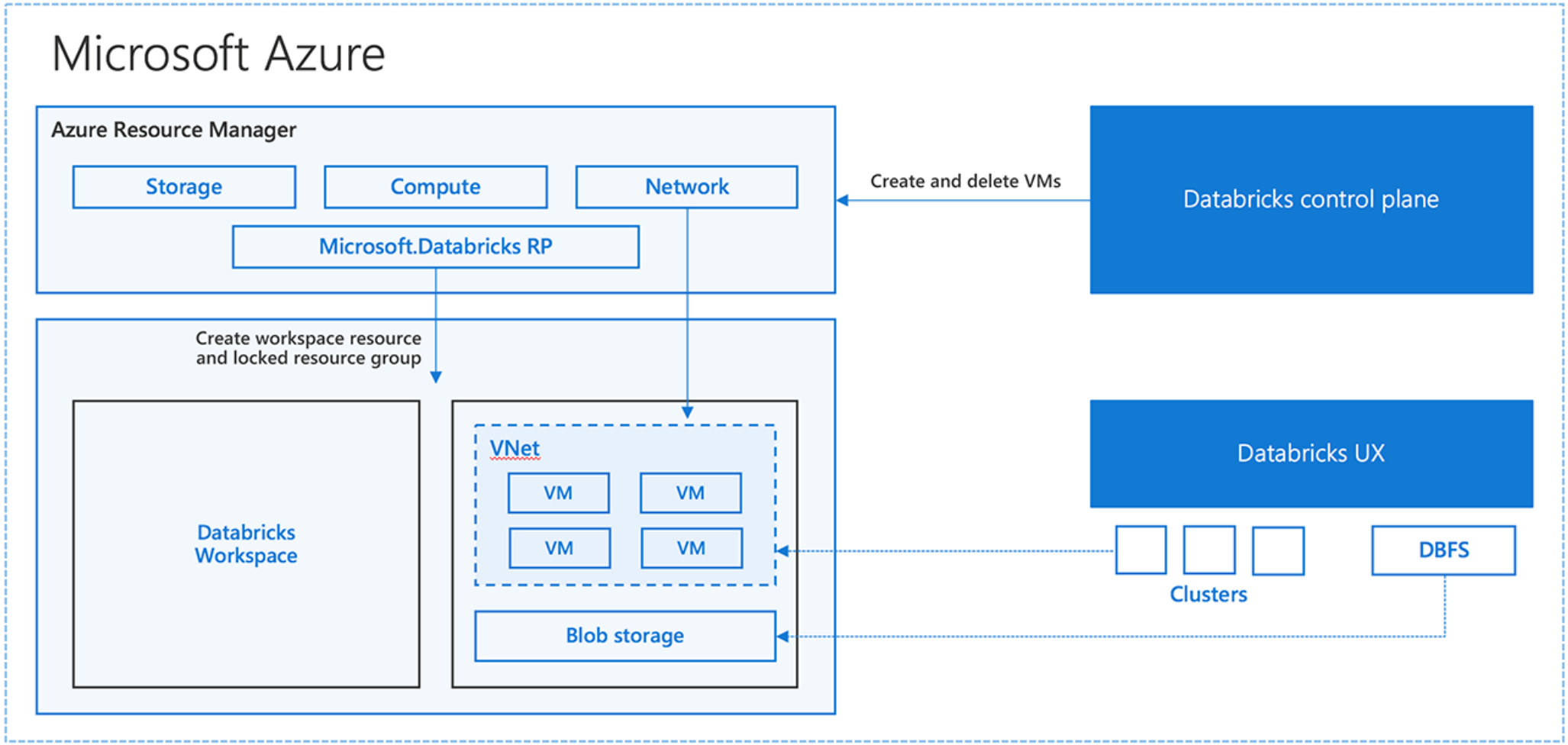
CETERIS AG

- Databricks is a company (original creators of Apache Spark)
- They offer a fast, easy and secure PaaS service to perform Spark operations



How is Databricks working in Azure?

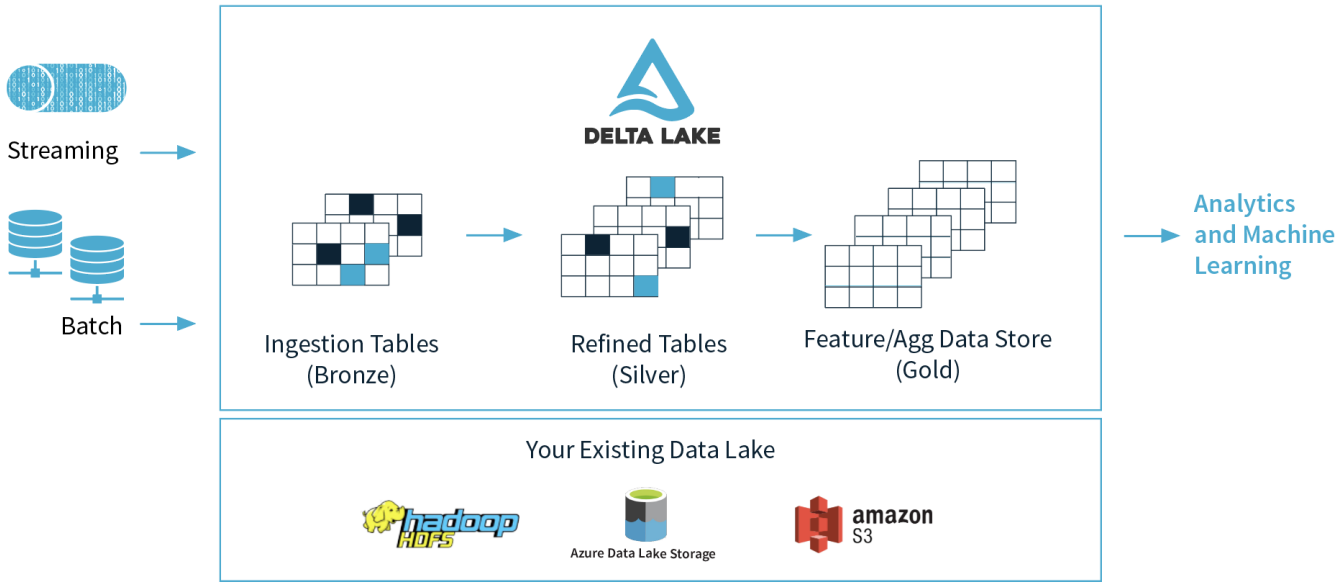
CETERIS AG



What is Delta Lake?

CETERIS AG

- Delta Lake is an open-source storage layer that brings ACID transactions and other relational database features to Apache Spark (on top of it).
- It provides:
  - ACID transactions
  - Time travel
  - Open-source storage format
  - Streaming sources and sinks
  - Schema enforcement as well as evolution
  - Audit History
  - Update / delete commands



How does Delta Lake ACID principle work?

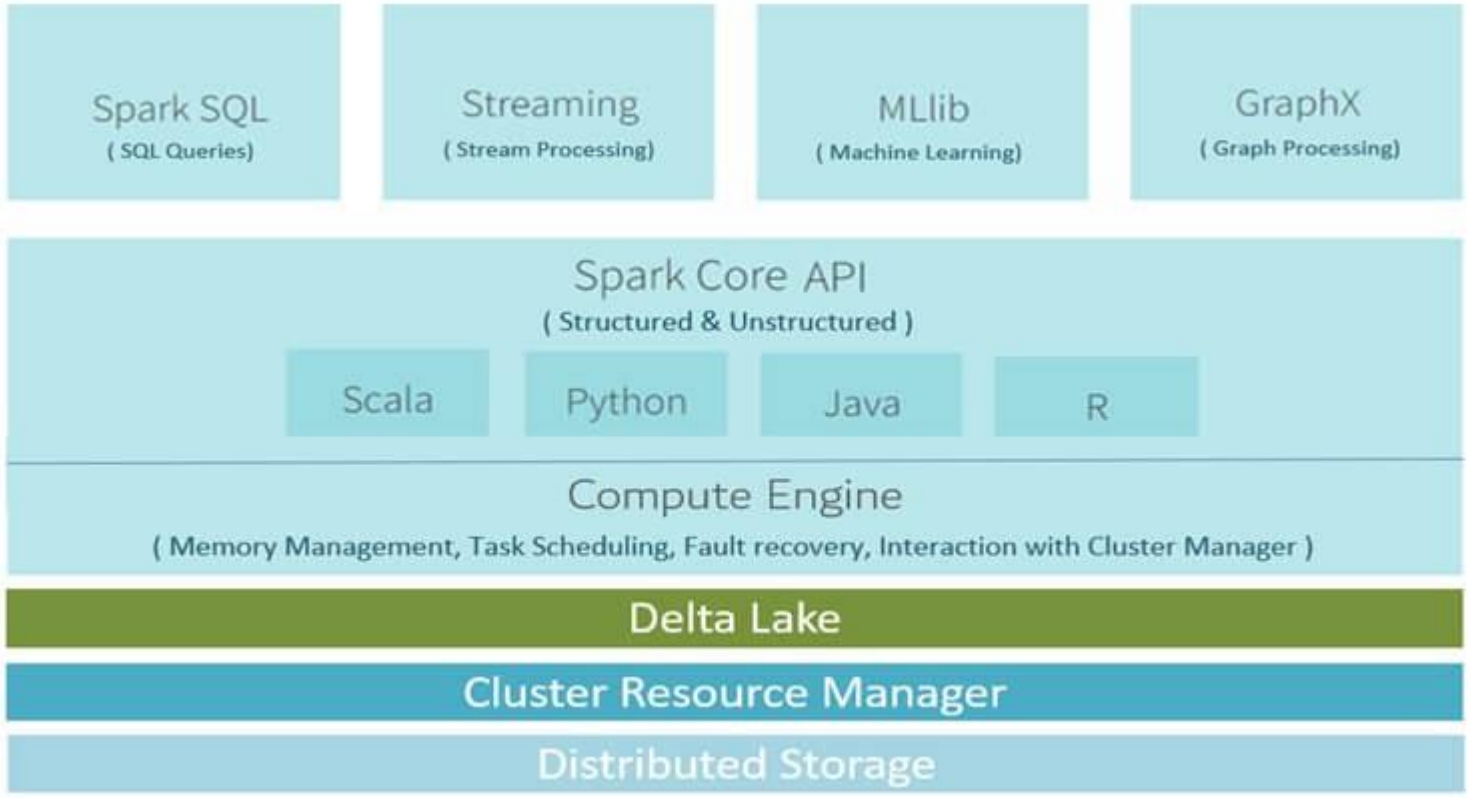
CETERIS AG

- Delta Lake guarantees atomicity and consistency via the so-called transaction log

  "*If it's not recorded in the transaction log, it never happened.*"

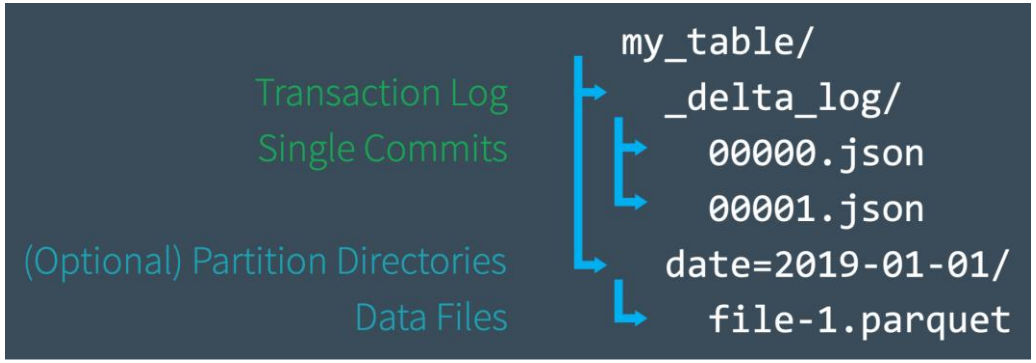
- It provides serializability as level of isolation

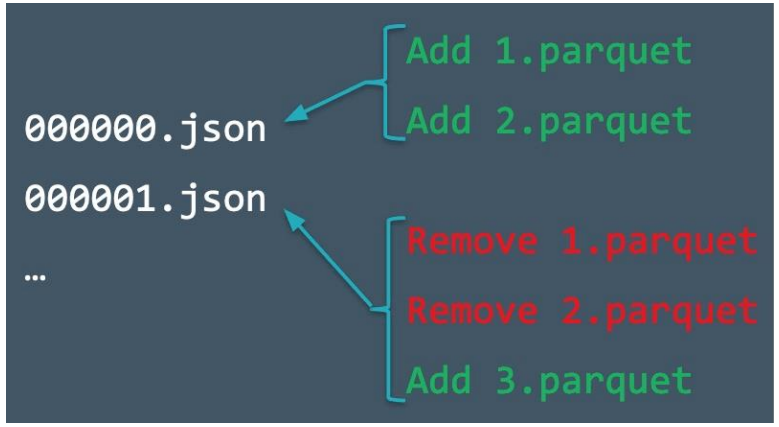
- Durability is automatically conserved due to all information being written directly to disk

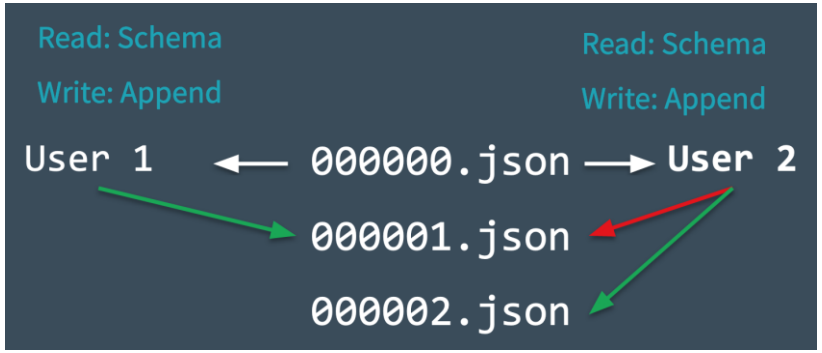


Transaction Log

CETERIS AG







Delta Lake 101

Create Table

CETERIS AG

Tables are just references and metadata

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

```
df.write.format("delta").saveAsTable("events")
```

Insert Into

CETERIS AG

```
INSERT INTO events SELECT * FROM newEvents
```

```
INSERT OVERWRITE events SELECT * FROM newEvents
```

```
df.write
  .format("delta")
  .mode("overwrite")
  .option("replaceWhere", "date >= '2017-01-01' AND date <= '2017-01-31'")
  .save("/mnt/delta/events")
```

Update / Merge

CETERIS AG

```scala
import io.delta.tables._

val deltaTable = DeltaTable.forPath(spark, "/data/events/")

deltaTable.updateExpr(            // predicate and update expressions using SQL for
  "eventType = 'clck'",
  Map("eventType" -> "'click'")

import org.apache.spark.sql.functions._
import spark.implicits._

deltaTable.update(                // predicate using Spark SQL functions and implic
  col("eventType") === "clck",
  Map("eventType" -> lit("click")));
```

```sql
MERGE INTO events
USING updates
ON events.eventId = updates.eventId
WHEN MATCHED THEN
  UPDATE SET events.data = updates.data
WHEN NOT MATCHED
  THEN INSERT (date, eventId, data) VALUES (date, eventId, data)
```

Delete

CETERIS AG

```
DELETE FROM events WHERE date < '2017-01-01'

DELETE FROM delta.`/data/events/` WHERE date < '2017-01-01'
```

```
import io.delta.tables._

val deltaTable = DeltaTable.forPath(spark, "/data/events/")

deltaTable.delete("date < '2017-01-01'")

import org.apache.spark.sql.functions._
import spark.implicits._

deltaTable.delete(col("date") < "2017-01-01")
```

Drop Table / Optimize / Vacuum

CETERIS AG

```
DROP TABLE IF EXISTS <example-table>     // deletes the metadata and data
CREATE TABLE <example-table> AS SELECT ...
```

```
DROP TABLE IF EXISTS <example-table>    // deletes the metadata
dbutils.fs.rm("<your-s3-path>", true)   // deletes the data
CREATE TABLE <example-table> USING org.apache.spark.sql.parquet
```

Traveling in time

„Time traveling? As if…“ – „AS OF“!

CETERIS AG

- Go back to the state of a table at a specific timestamp or table version
- Scala/Python: spark.read.(…).option(„timestampAsOf“,“2020-07-02“).load(„myPath“)
- SQL: SELECT * FROM myTable VERSION AS OF 1
  - View table versions and audit information with DESCRIBE HISTORY (or just use the UI)
- Use Cases: Rollbacks, time series analytics, pinned views,…

(Unfortunately, you can really just go to the past and back to the future, aka the present)

Creating and writing streams

Creating a stream from different sources...

CETERIS AG

- Get a Databricks cluster up and running (and add any configs and libraries before you start it up)
- Before you stream anything to delta, configure your Gen2 storage and a mounting point
- Think about creating „external" tables (i.e. not managed by Databricks) beforehand
- Prepare source configuration
  - File names/locations
  - EventHub endpoint
  - SQL Server jdbc drivers
  - …

… and write it to a delta table

CETERIS AG

- Basic scala syntax:

```
insertDF.writeStream
  .format("delta")
  .outputMode("append")
  .option("checkpointLocation", "/mnt/MountFolder/TableName/_checkpoints/etl-from-json")
  .start("/mnt/MountFolder/TableName") //or table(„TableName„)
  .trigger(Trigger.Once)
  .partitionBy(„PartitionColumn1",„PartitionColumn2")
```

- Output options:
  - append – default, appends rows to existing or newly created table
  - complete – replace the entire table
  - update – only writes rows that have changed since last trigger (only used with aggregations)
- Trigger options:
  - Trigger.Once – triggers exactly once and then stops the stream (in theory…)
  - Trigger.ProcessingTime("60 seconds") – triggers in given interval (can be anything from ms to days)
  - Default: behaves as if ProcessingTime set to 0 ms, tries to fire queries as fast as possible

Power BI Visualization on Delta Tables

How to connect to Databricks?

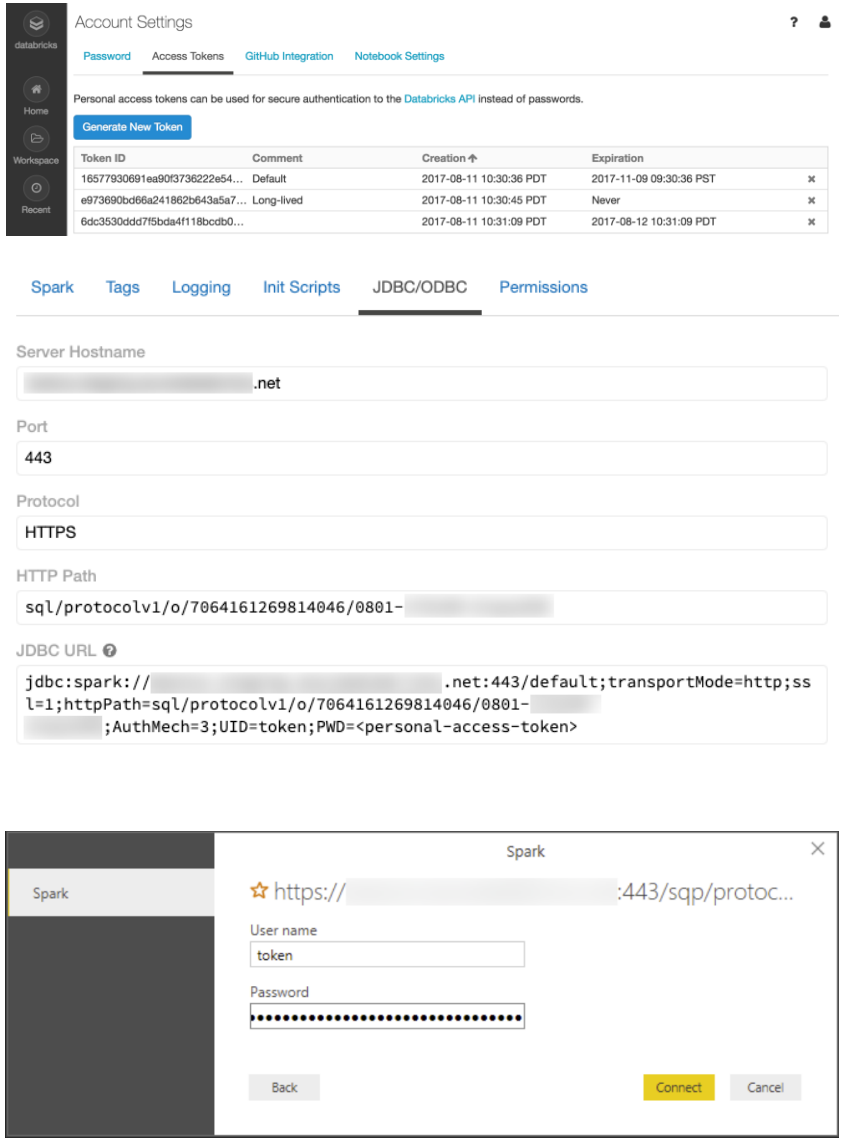
CETERIS AG

1. Get a personal access token
2. Get your cluster's server hostname, port, and HTTP path
3. Construct the server address to use in in Power BI Desktop
   a. Use the schema https://
   b. Append the server hostname after the schema
   c. Append the HTTP path after the server host name

→ https://westeurope.azuredatabricks.net/sql/protocolv1/o/0123456789/0123-456789-sometext

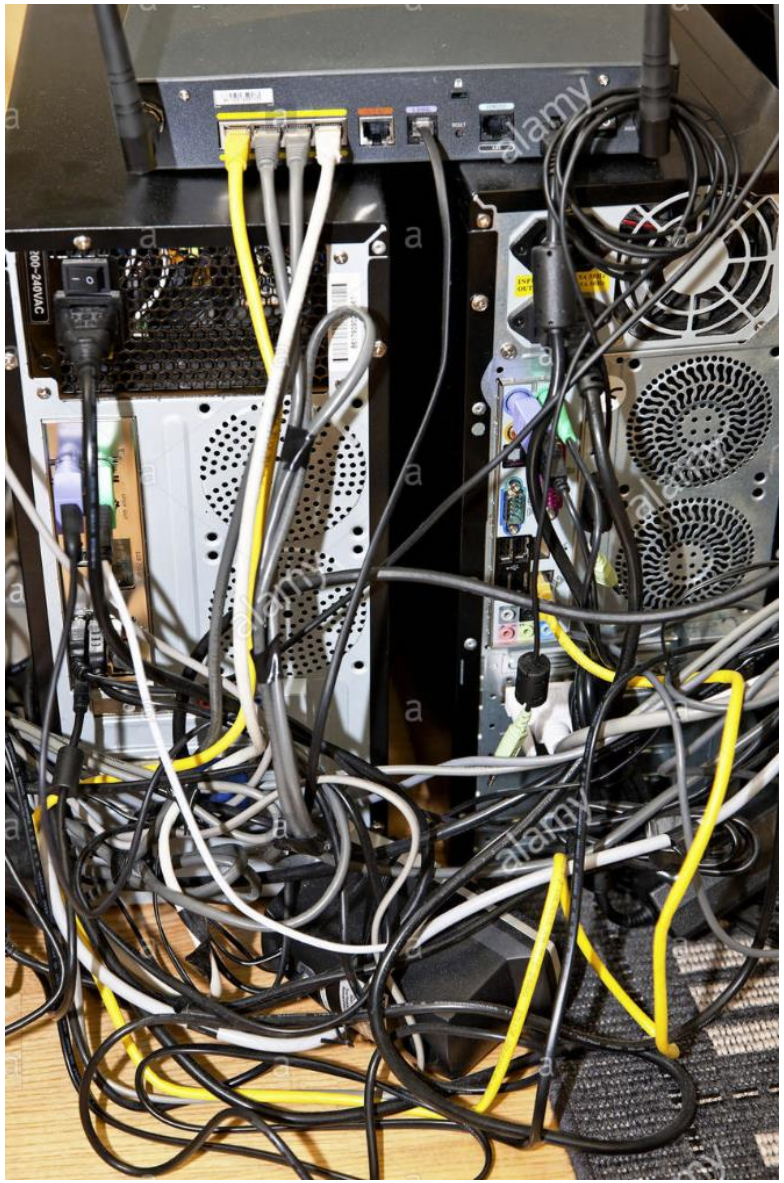
4. In Power BI use Spark connector and use
   a. „token" as username
   b. personal access token as password

It supports Import AND DirectQuery models!



How to connect to Databricks?

CETERIS AG



Streaming DWH Demo

What this demo will show (if there's enough time)

CETERIS AG

- Streaming from ~~EventHub~~ storage all the way to Synapse
- Joins in streams
- Watermarking
- How to actually write to tables in Synapse and why we ended up needing a classic blob storage
- Including user-defined functions
- Handling late-arriving data and SCD2

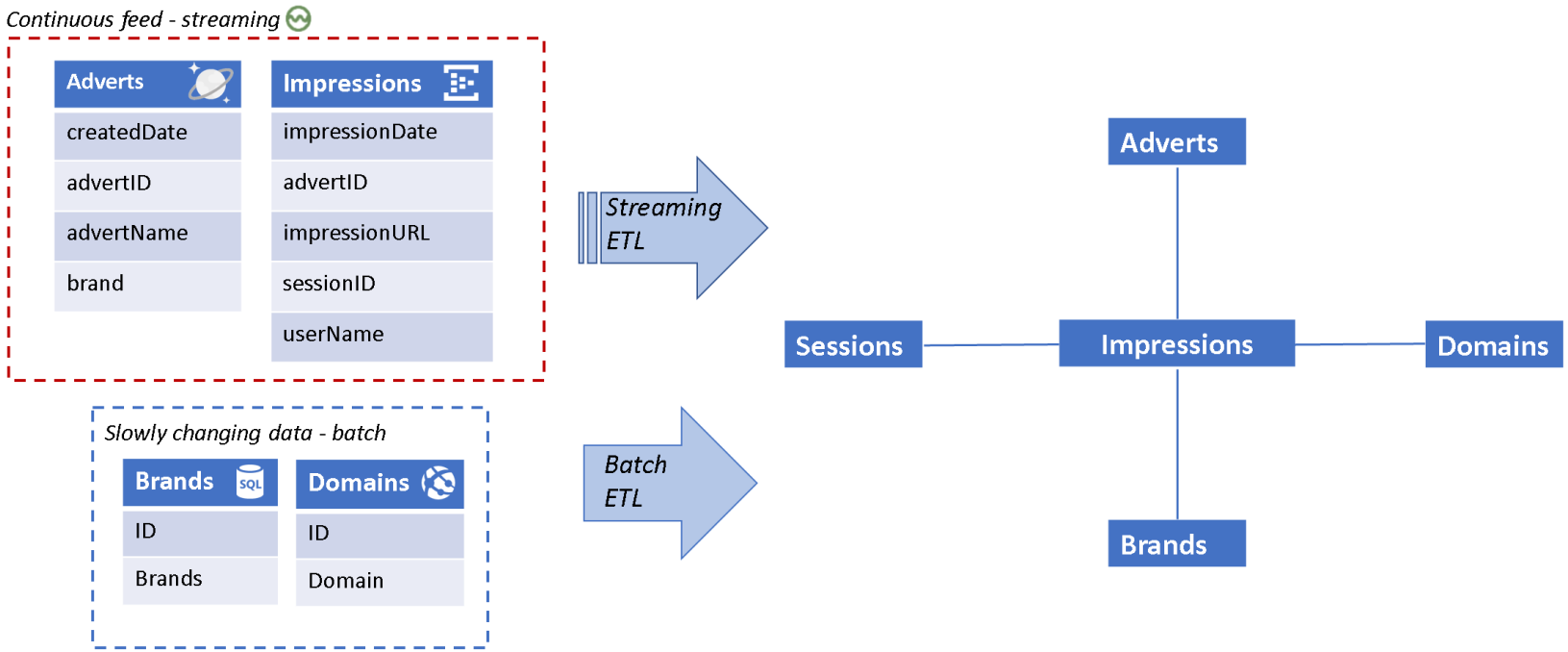
Scenario and source

CETERIS AG

- Demo available by following instructions on blog by Nicholas Hurt:

https://medium.com/microsoftazure/an-introduction-to-streaming-etl-on-azure-databricks-using-structured-streaming-databricks-16b369d77e34

- …though of course we had to simplify and change it up a bit

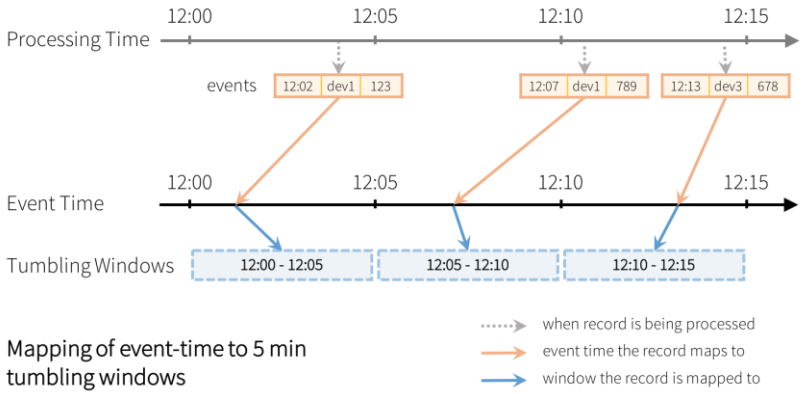


Windows

CETERIS AG

- Define windows for aggregations
  - Tumbling window

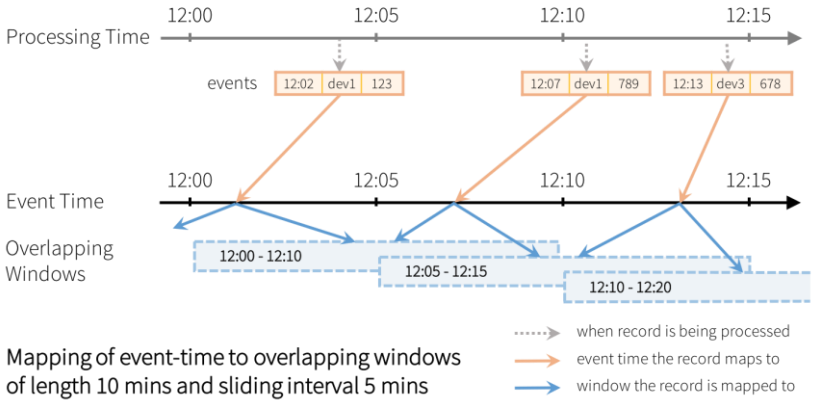
```
windowedAvgSignalDF = \
  eventsDF \
    .groupBy(window("eventTime", "5 minutes")) \
    .count()
```



Mapping of event-time to 5 min tumbling windows

  - overlapping window

```
windowedAvgSignalDF = \
  eventsDF \
    .groupBy(window("eventTime", “10 minutes", "5 minutes")) \
    .count()
```



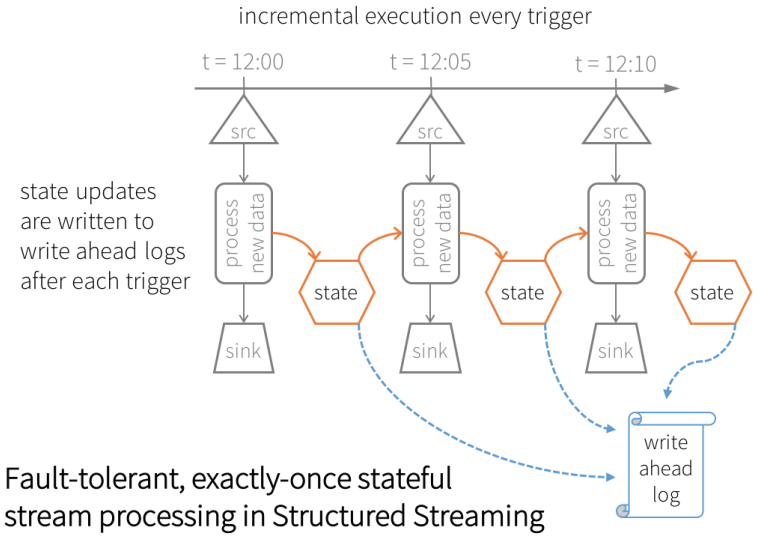
Mapping of event-time to overlapping windows of length 10 mins and sliding interval 5 mins

Source:
https://databricks.com/blog/2017/05/08/event-time-aggregation-watermarking-apache-sparks-structured-streaming.html

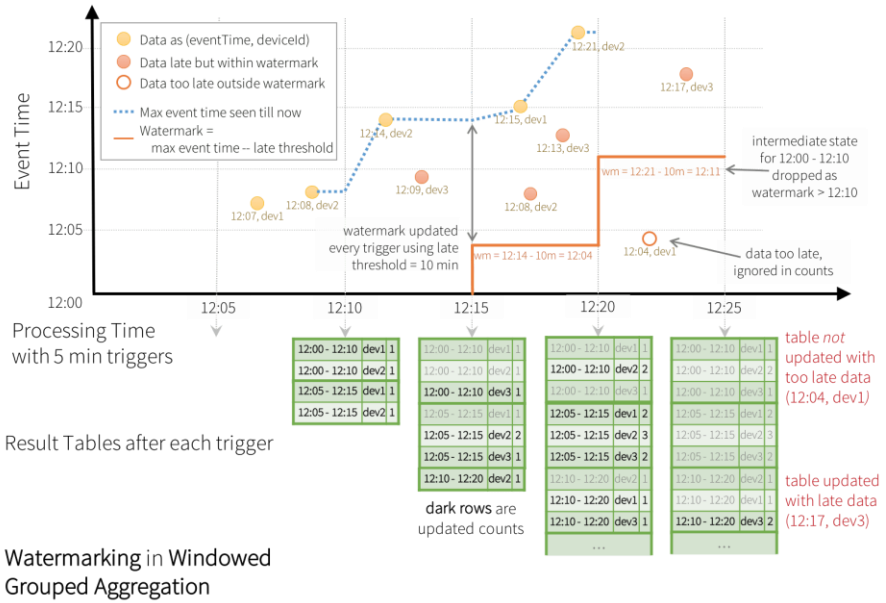
Watermarking

CETERIS AG

- States:



- How to prevent inevitable memory overflow?
  - .withWatermark(„eventTime“,“10 minutes“)



Source:
https://databricks.com/blog/2017/05/08/event-time-aggregation-watermarking-apache-sparks-structured-streaming.html

Joins in delta streams

CETERIS AG

- Due to data continuously streaming, joins have to be reimagined a bit
- For every row of table a, there could always arrive a row in table b that matches
- For left joins, you cannot ever tell if there will be no match and result has to be NULL
- Solution: use watermarks to determine how long the join operation waits for matches!
  - Define watermark delays on inputs
  - Define event-time range conditions on join operator

```
// Define watermarks
val impressionsWithWatermark = impressions
  .select($"adId".as("impressionAdId"), $"impressionTime")
  .withWatermark("impressionTime", "10 seconds ")   // max 10 seconds late

val clicksWithWatermark = clicks
  .select($"adId".as("clickAdId"), $"clickTime")
  .withWatermark("clickTime", "20 seconds")        // max 20 seconds late


// Inner join with time range conditions
display(
  impressionsWithWatermark.join(
    clicksWithWatermark,
    expr("""
      clickAdId = impressionAdId AND
      clickTime >= impressionTime AND
      clickTime <= impressionTime + interval 1 minutes
      """
    )
  )
)
```

Questions?

CETERIS AG

thuss@ceteris.ag

tsalha@ceteris.ag

LinkedIn:

https://www.linkedin.com/in/thorstenhuss/

https://www.linkedin.com/in/tarek-salha-2a39ab189/

Web: www.ceteris.ag