# THE DATA SCIENCE LAB
# Introduction to Data Stream Processing

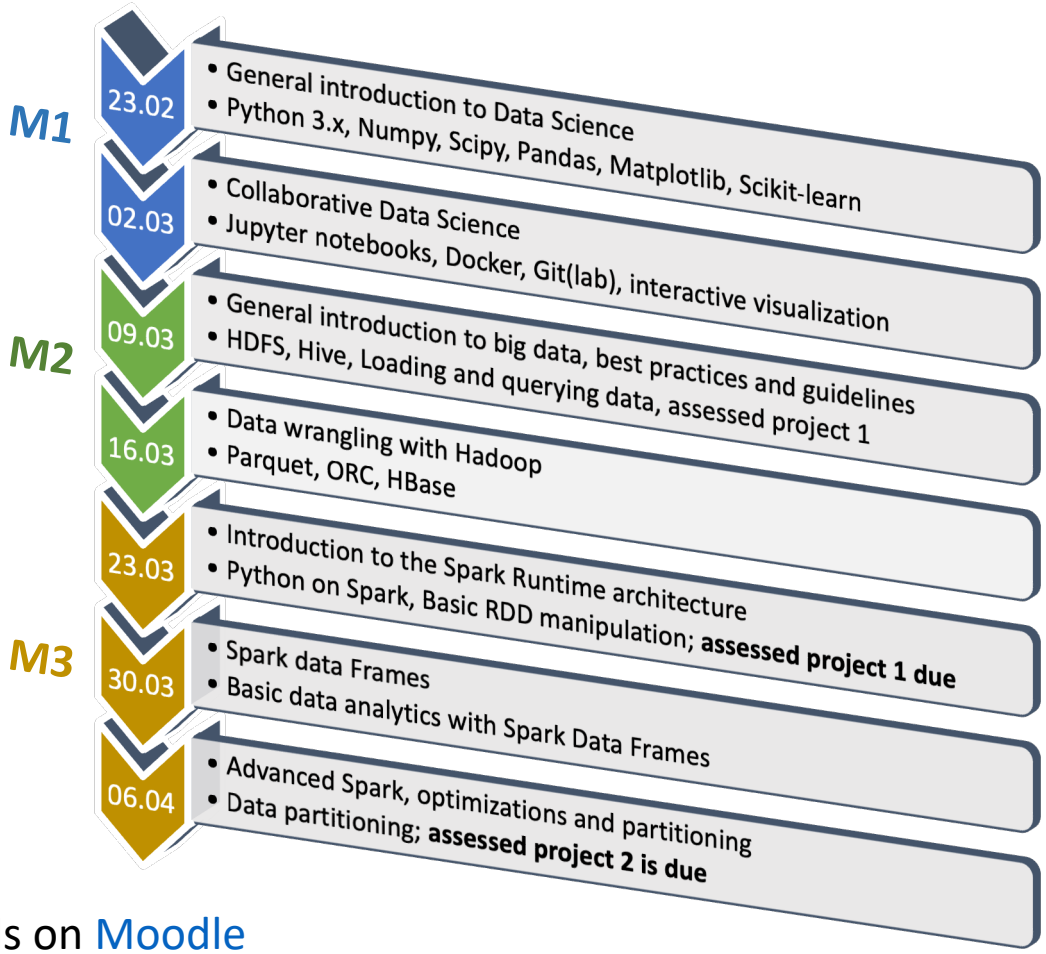
COM 490 – Spring 2022

Week 10

**THIS CLASS WILL BE RECORDED**

EPFL

# Agenda Spring 2022





*Details on Moodle

# Today

- Stream Processing
  - Advanced topics
    - Operations on streaming data (joins)
    - Time constraints
- Exercises (demonstration)
  - https://dslab2022-renku.epfl.ch/projects/com490/lab-course
  - *Go to week 9*
- Homework 4
  - https://dslab2022-renku.epfl.ch/projects/com490/homework-4
  - due before 10.05 23h59
- Final project instructions

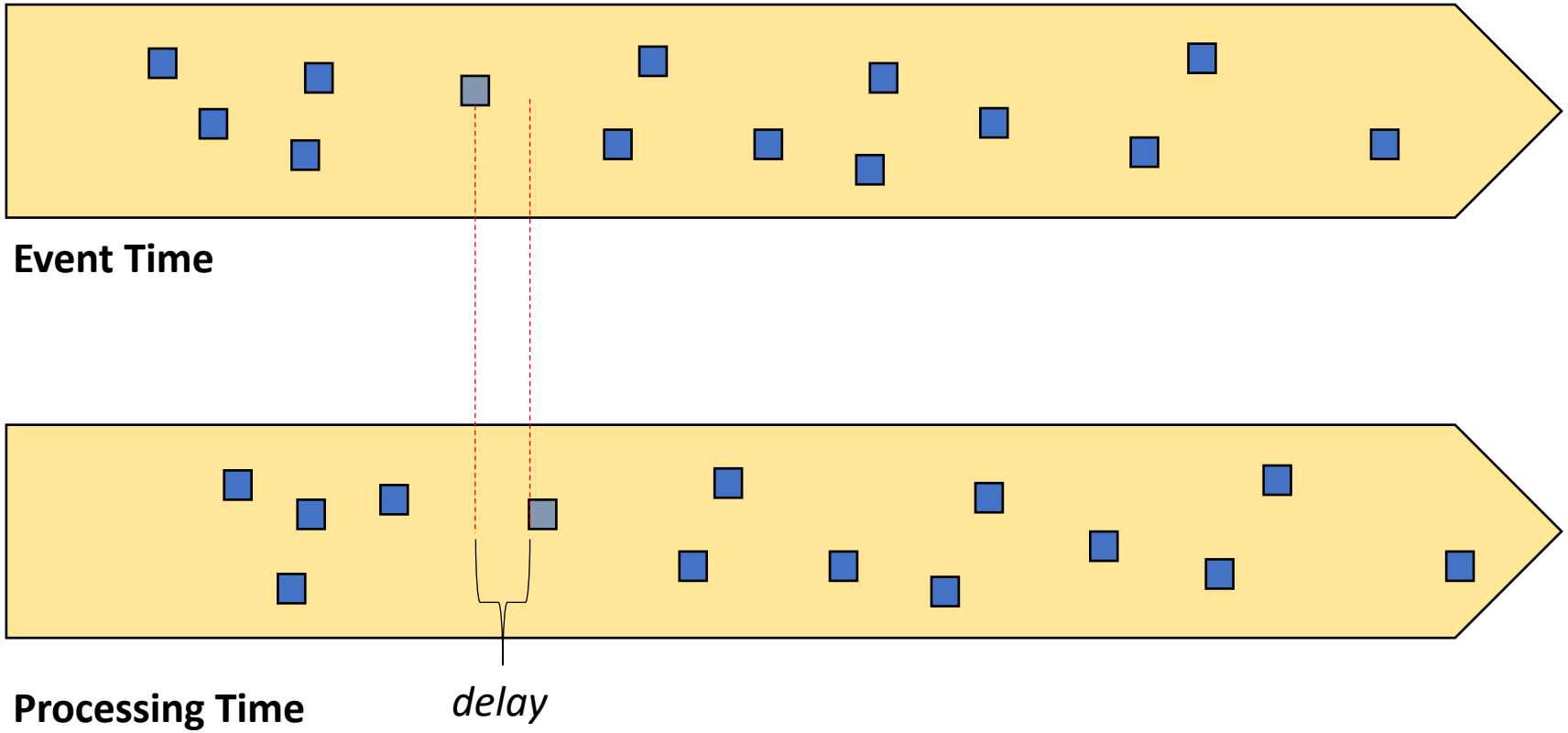
# Stream Processing Module

- Objectives
  - Review concepts of stream processing
  - Experiment with typical tools for
    - Data ingestion and processing
- Week 8
  - Concepts
  - Experiments
- Week 9
  - Advanced topics
    - Operations on streaming data (joins)
    - Time constraints
- Week 10
  - Analytics on data at rest and data in motion
  - Graded homework

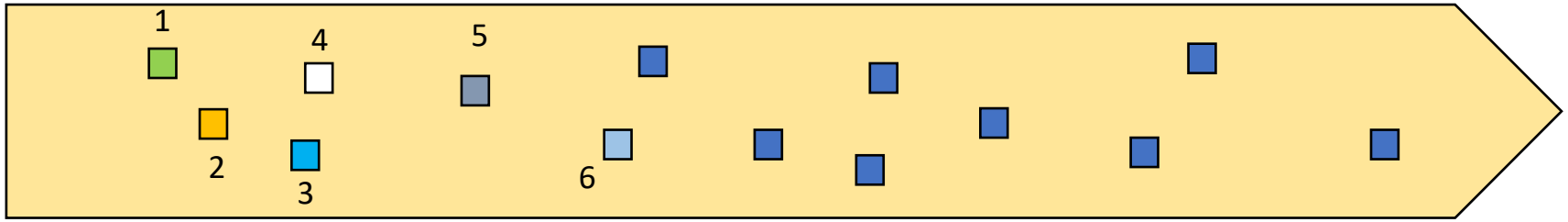
# Today's Agenda

- Week 10
  - Advanced topics
  - Solution to exercises
- Introduction of the final project
- Your questions
  - exercises
  - homework 4

# Event Time vs Processing Time

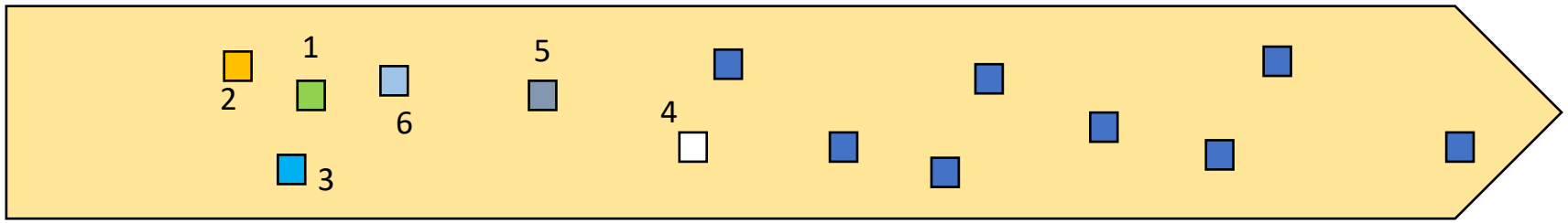


# Event Time vs Processing Time




**Event Time**

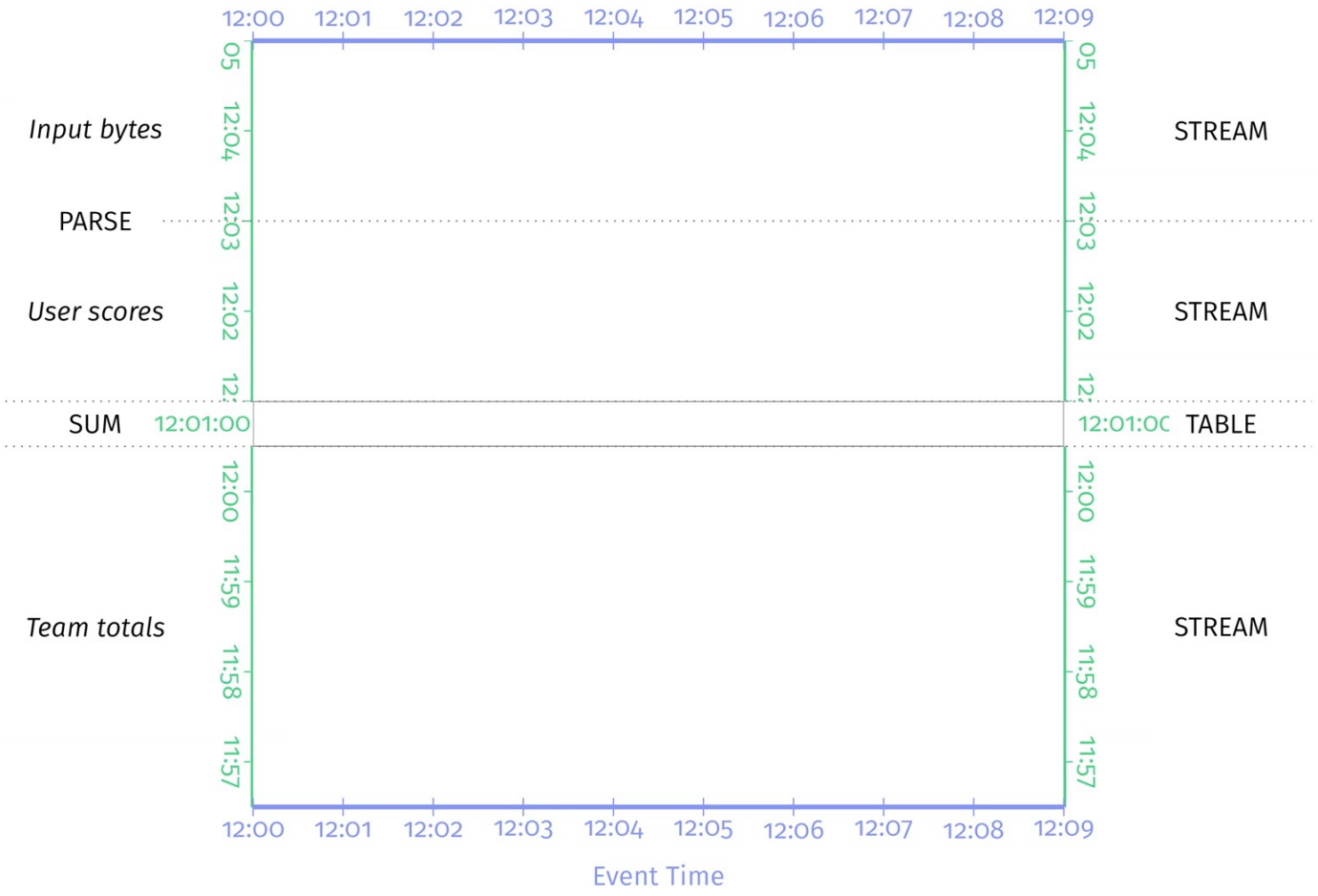
1, 2, 3, 4, 5, 6, …




**Processing Time**

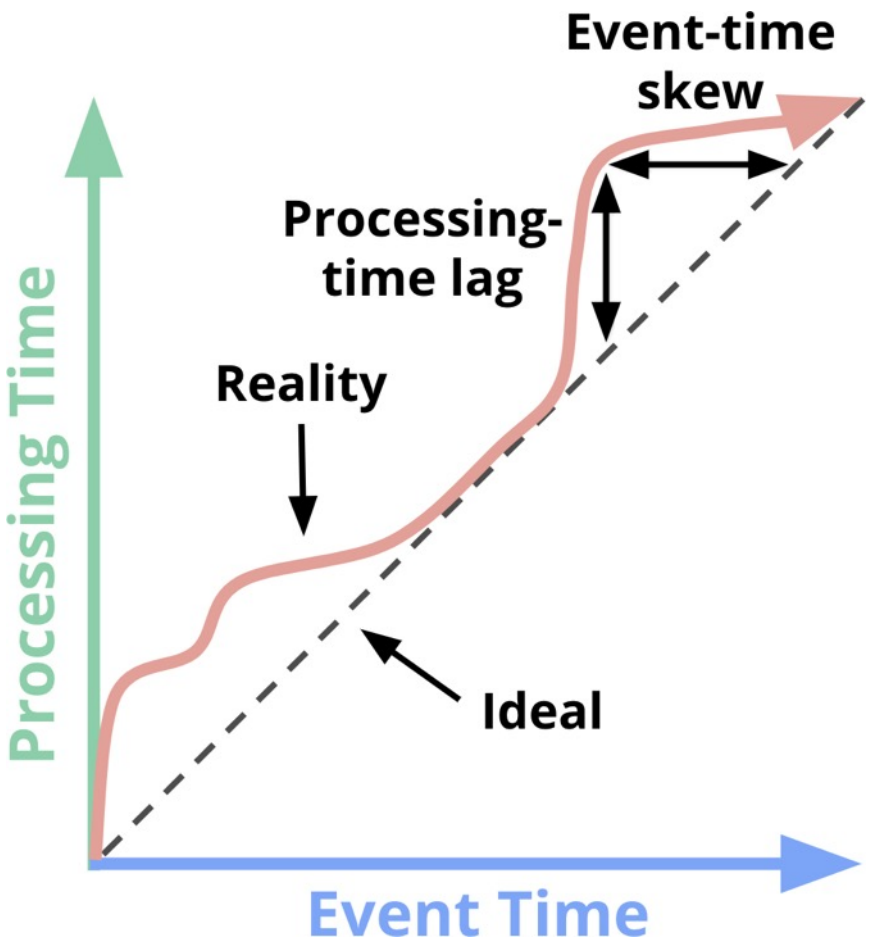
2, 3, 1, 6, 5 , 4, …

# Event Time vs Processing Time



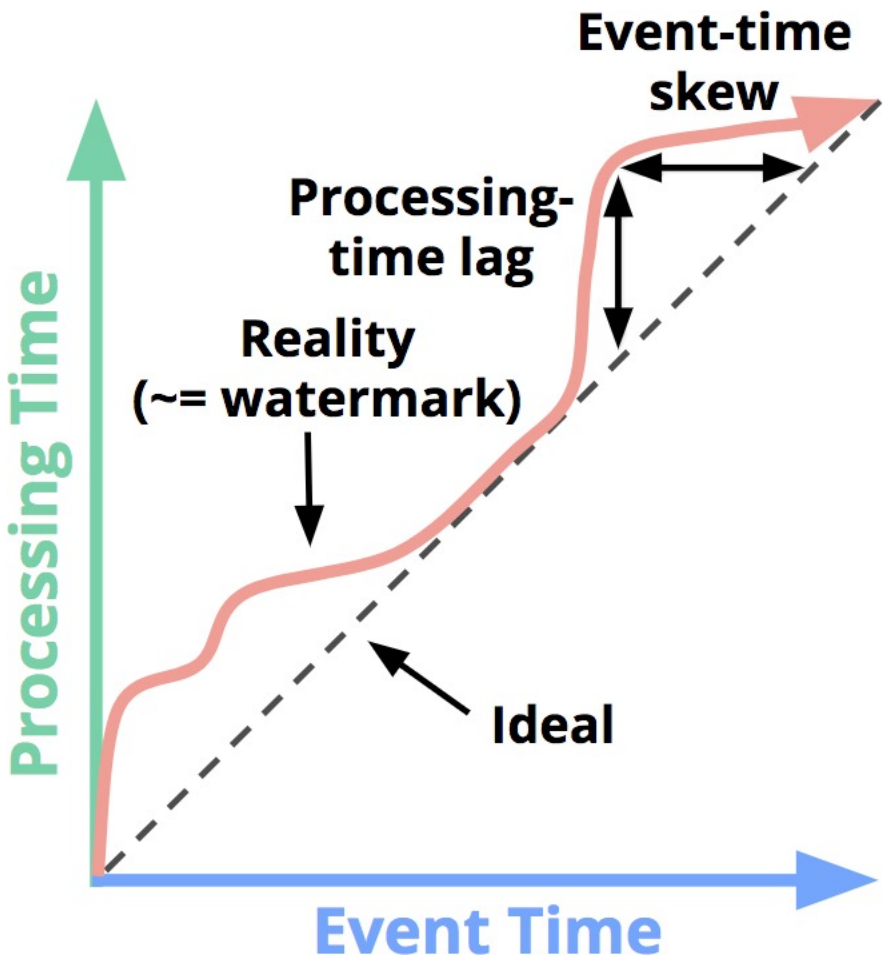

Credits: Tyler Akidau (et al.), Streaming Systems, O'Reilly Media, 2018.

# Event Time vs Processing Time




Credits: Tyler Akidau (et al.), Streaming Systems, O'Reilly Media, 2018.

# Event Time vs Processing Time




Credits: Tyler Akidau (et al.), Streaming Systems, O'Reilly Media, 2018.

# Watermarking in Spark Streaming

Count on a sliding window of 10 minutes long with 5 minutes sliding interval with 10 minutes watermarking (lateness).

```
from pyspark.sql.functions import *


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

# Operations

- How it works

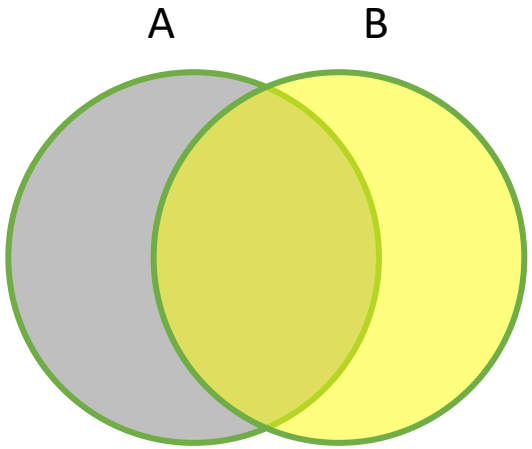

- DStream: continuous stream of data
  - Created from inputs (e.g. Kafka) or derived from other DStreams
  - Supports transformations like RDDs
    - (map, count, join, etc)

Image credits: https://spark.apache.org/docs/latest/streaming-programming-guide.html

# Operations: Joins

- Joins
  - Combining different sets of data (tables) to get a set of results based on some criteria

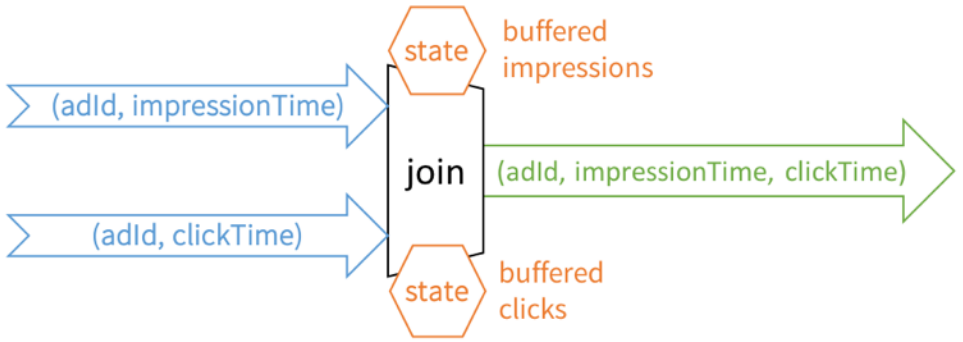
- Streaming Streaming
  - Stream-Static joins
  - Stream-Stream joins



# Operations: Joins

- Stream-Stream Joins:
  - 2 streams over a common key

- Example: Ad monetization (*)

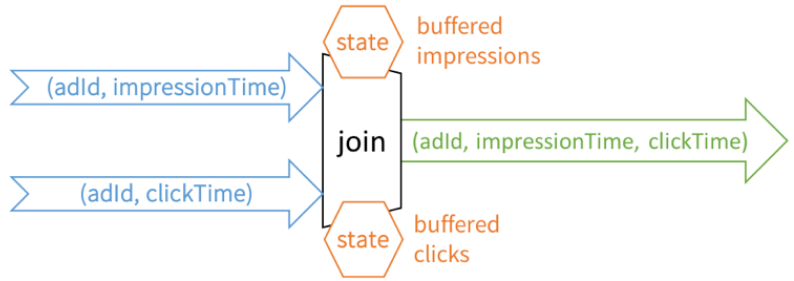



Stream-stream join use case: Ad-Monetization (joining ad clicks to impressions)

*(*) Databricks Engineering Blog, Introducing Stream-Stream Joins in Apache Spark 2.3 by Tathagata Das and Joseph Torres , https://bit.ly/2I58Ve7*

# Operations: Joins

- Streaming Constraints (*)
  - Handling of late/out of order data
  - Boundaries




Stream-stream join use case: Ad-Monetization (joining ad clicks to impressions)

- Solution
  - Use watermarks for delays / out of order impressionTime and clickTime
  - Set time range boundaries for the join conditions on event-time windows

*(*) Databricks Engineering Blog, Introducing Stream-Stream Joins in Apache Spark 2.3 by Tathagata Das and Joseph Torres , https://bit.ly/2I58Ve7*

# Operations: Joins

```
from pyspark.sql.functions import expr

impressions = spark.readStream. ...
clicks = spark.readStream. ...

# Apply watermarks on event-time columns
impressionsWithWatermark = impressions.withWatermark("impressionTime", "2 hours")
clicksWithWatermark = clicks.withWatermark("clickTime", "3 hours")

# Join with event-time constraints
impressionsWithWatermark.join(
  clicksWithWatermark,
  expr("""
    clickAdId = impressionAdId AND
    clickTime >= impressionTime AND
    clickTime <= impressionTime + interval 1 hour
    """)
)
```

# Useful references


[1] https://spark.apache.org/docs/latest/streaming-programming-guide.html

[2] https://kafka.apache.org/documentation/

[3] https://www.oreilly.com/radar/the-world-beyond-batch-streaming-101/

[4] http://www.streamingbook.net/