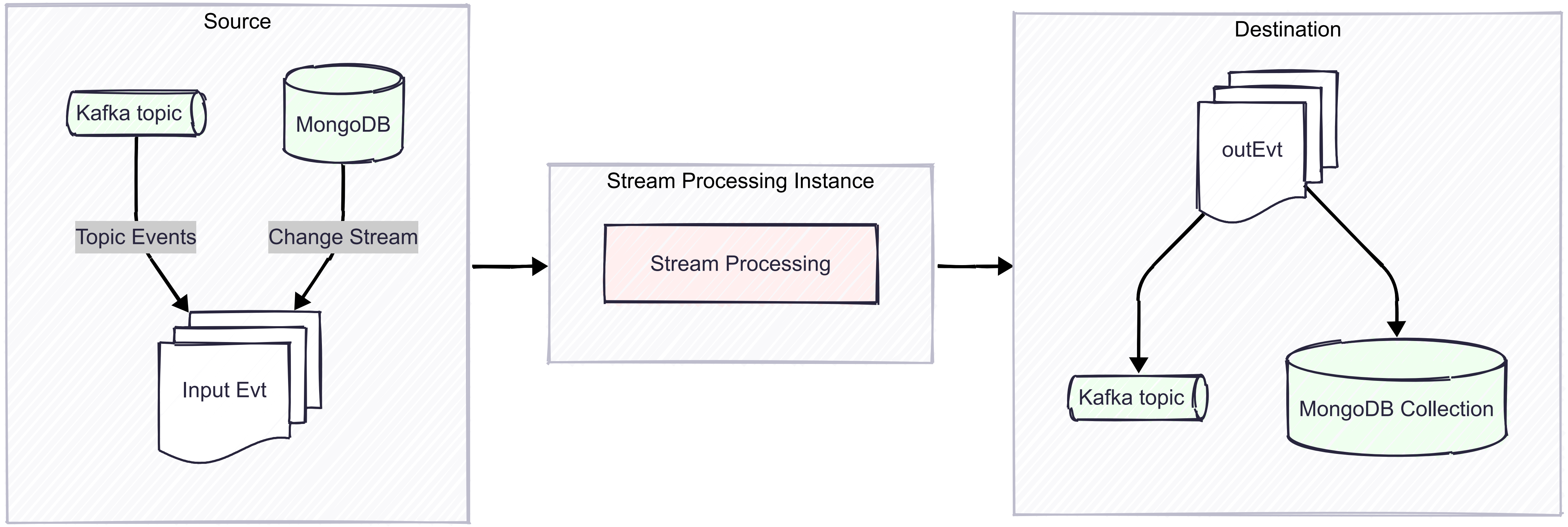
Events at the Movies with Atlas Stream Processing

A talk about streams of events, and making sense of them.

Why?

- Event Driven Architecture
    - Lots of events, sporadic
    - Common integration pattern, existing
- Stream Analytics
    - Want to learn "what's going on"
    - "Recent Analytics", without the hotspots
    - No extra durable storage.

Processing as a Pipeline



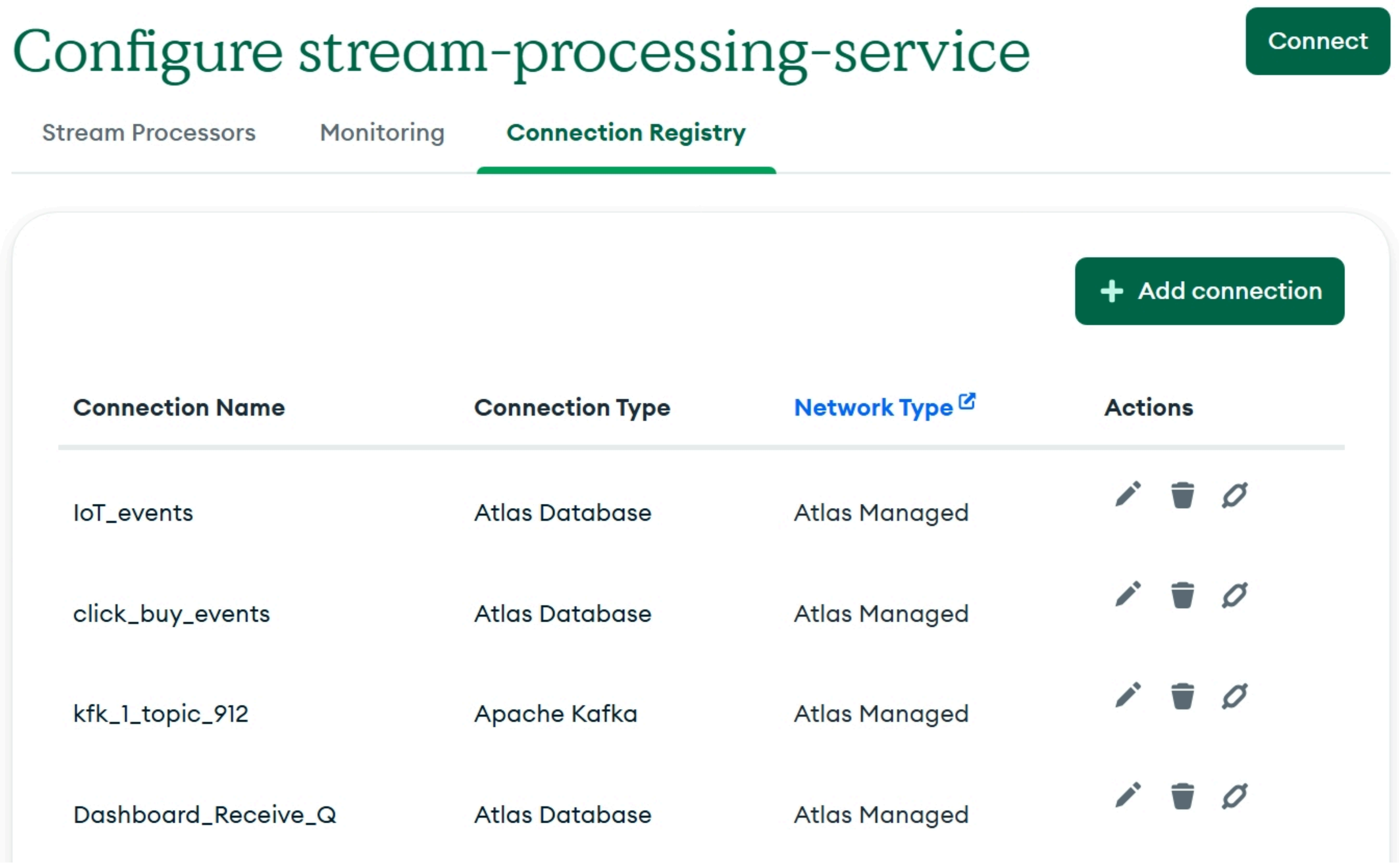
> Stream processor consumes events, and produces documents

In / Out

Connection Registry

- Kafka
- Atlas Database
- S3

  ...



Configure stream-processing-service

Connection Name	Connection Type	Network Type	Actions
IoT_events	Atlas Database	Atlas Managed	
click_buy_events	Atlas Database	Atlas Managed	
kfk_1_topic_912	Apache Kafka	Atlas Managed	
Dashboard_Receive_Q	Atlas Database	Atlas Managed	

Stream Processor Connections

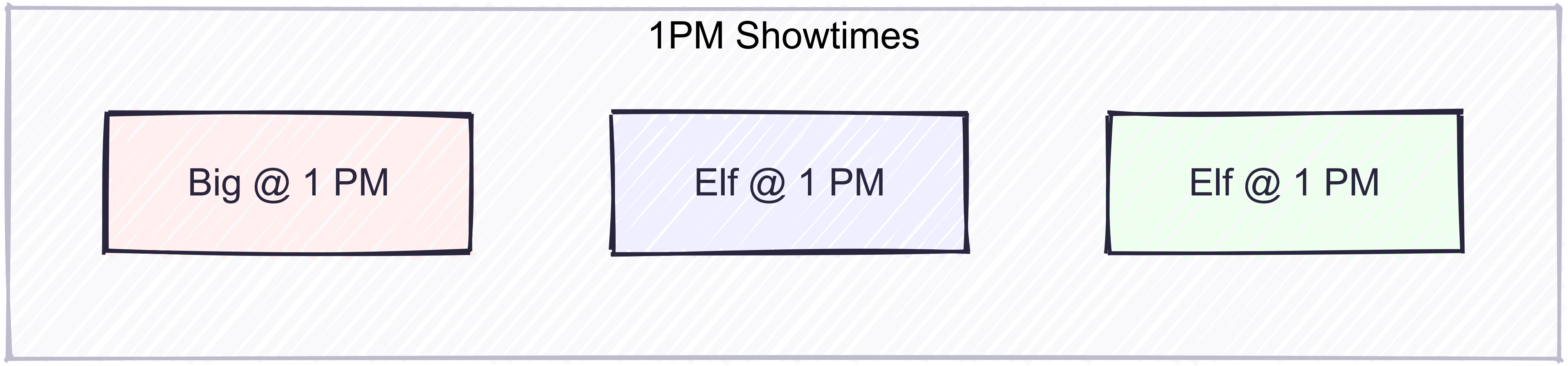
`connectionName` as configured

- 1st stage: `$source`
- last state: `$merge` | `$emit |` `$externalFunction`

```
[
  { $source: {
      connectionName: "mdbIn",
      db: "stream-demo",
      collection: "things" }},
  // { some processing stages...},
  { $merge:{
      into:{
        connectionName:"mdbConn",
        db:"db1",
        coll:"c1"} } }
]
```

Windowing and time basis

Windows are fixed width (usually).



Output is computed on events within its time boundaries.

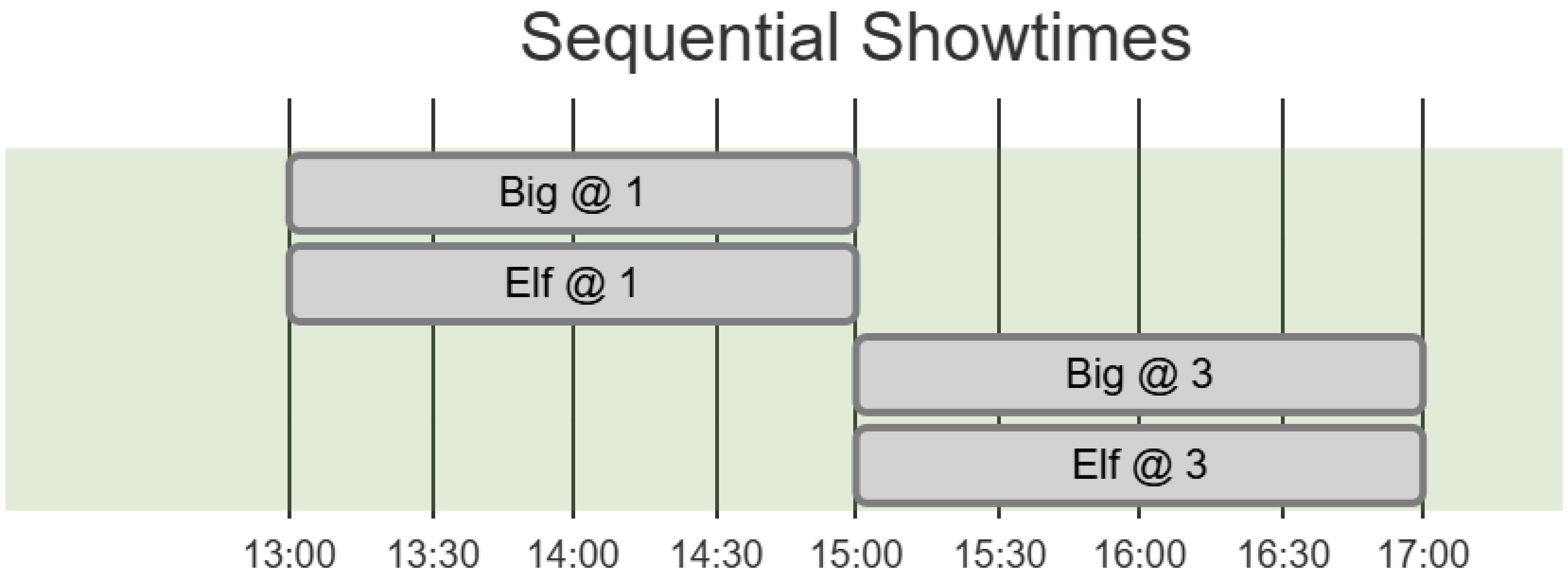
Create Stream Processor - How?

```
const pipeline = [{$source: ...}, ...];

sp.createStreamProcessor("mySP", pipeline)
```

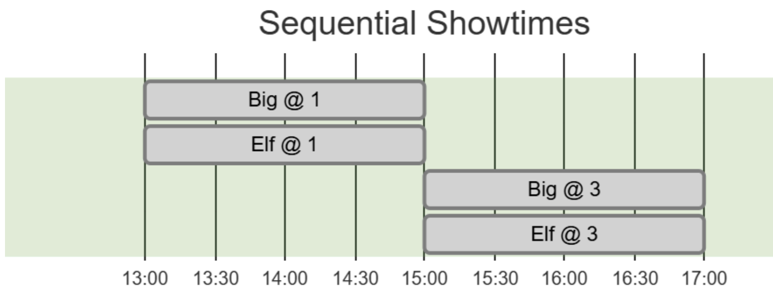
- `pipeline` always starts with a `$source` stage.

Tumbling Window

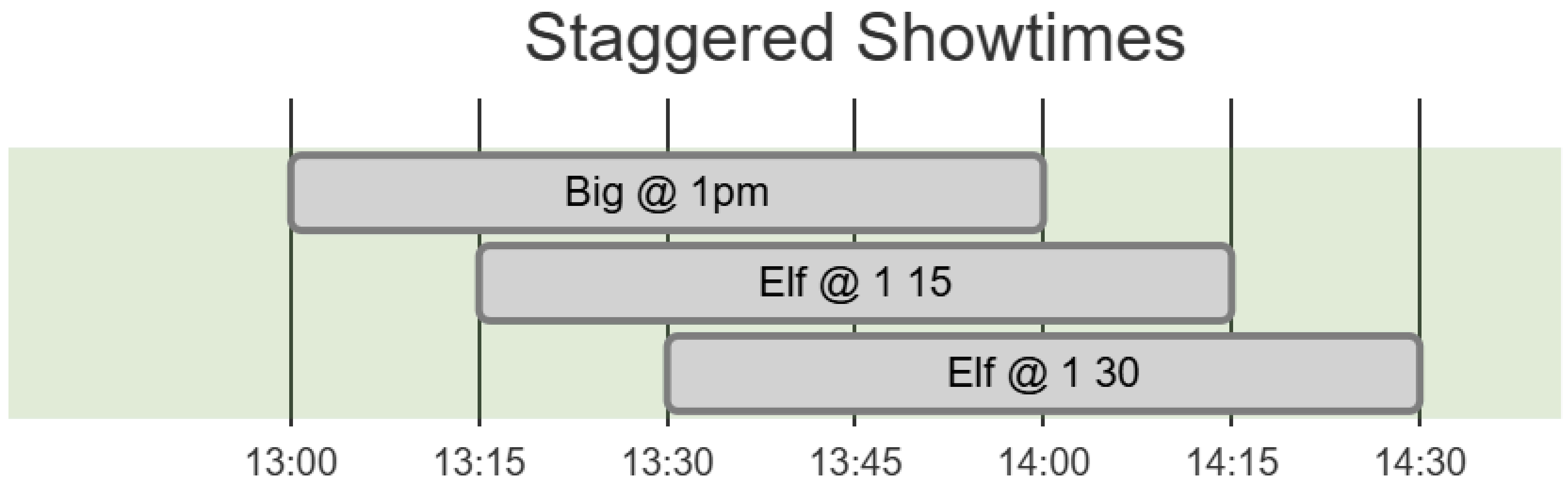


Tumbling - How?

```
{
  $tumblingWindow: {
    interval: { size: 30, unit: "seconds" },
    pipeline: [
      {
        $group: {
          _id: "$movie",
          total: { $sum: "$amountPaid" }
        }
      }
    ] ...
```

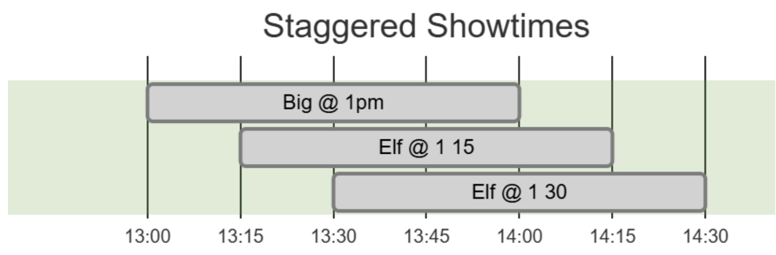


Hopping Window



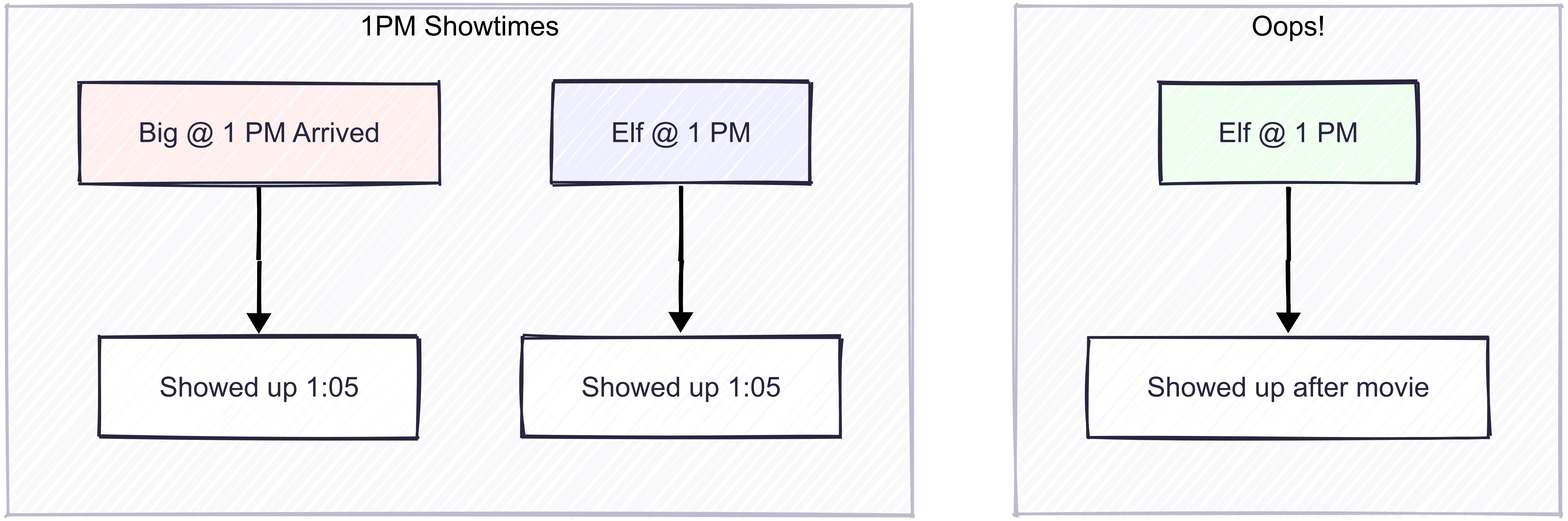
Hopping - How?

```
{
  $hoppingWindow:
  {
    interval: {size: 20, unit: "minute" },
    hopSize:  {size: 10, unit: "minute" },
    pipeline: [
      {
        $group: {
          _id: "$movie",
          walkIns: { $sum: "$ticketCount" }
        }
      }
    ] ...
```



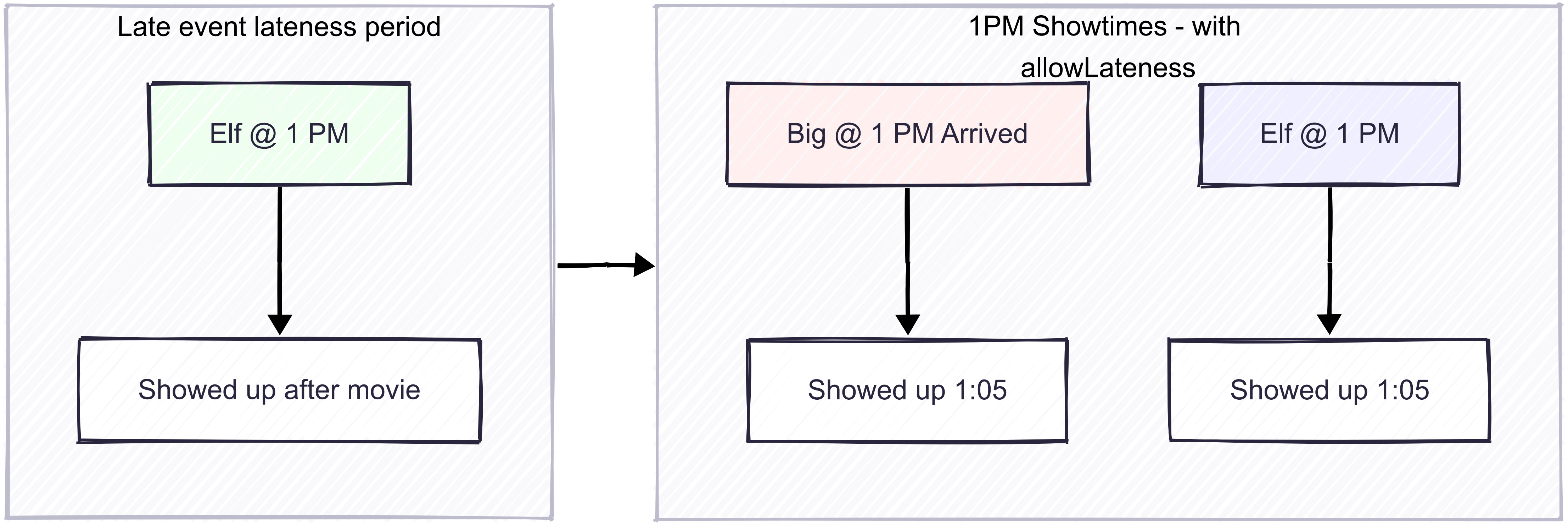
Missed the Window

Oops! What to do?

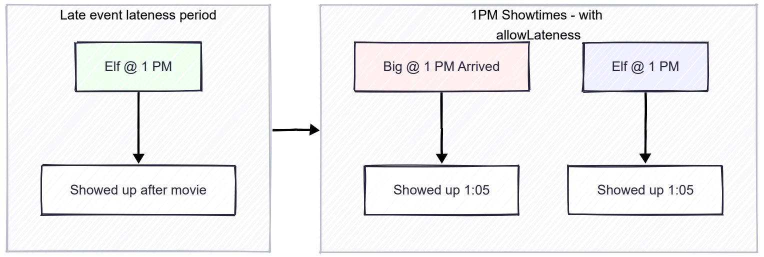


Allowed Lateness

`allowedLateness` lets late arrivals to be counted after *window-end-time*.



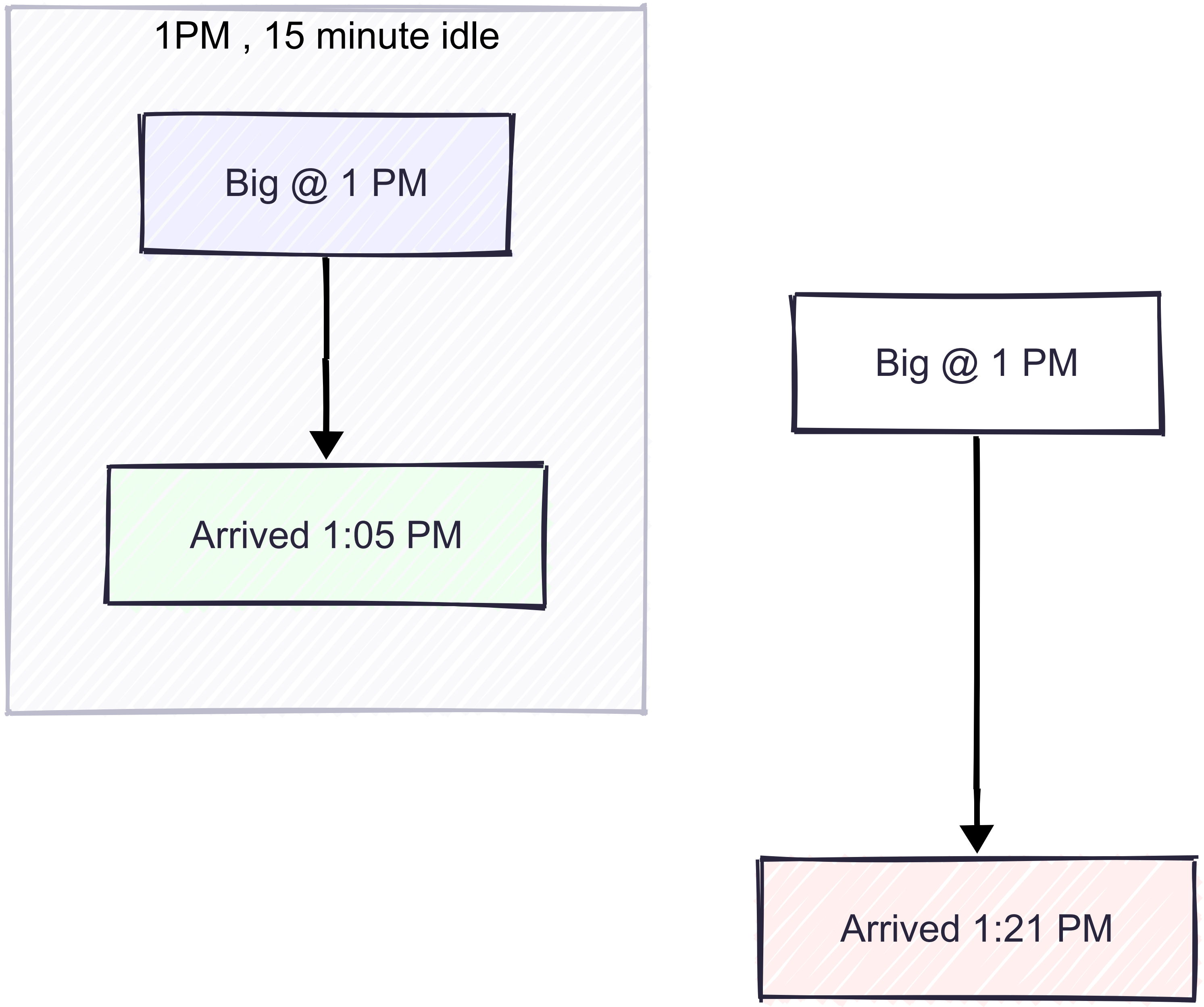
Lateness - How?



```
{
  $tumblingWindow: {
    allowedLateness: { size: 1, unit: "minute"},
    interval: { size: 30, unit: "seconds" },
    pipeline: [
      { $group: {
          _id: "$movie",
          total: { $sum: "$amountPaid" }
        } }
    ]
  }
}
```

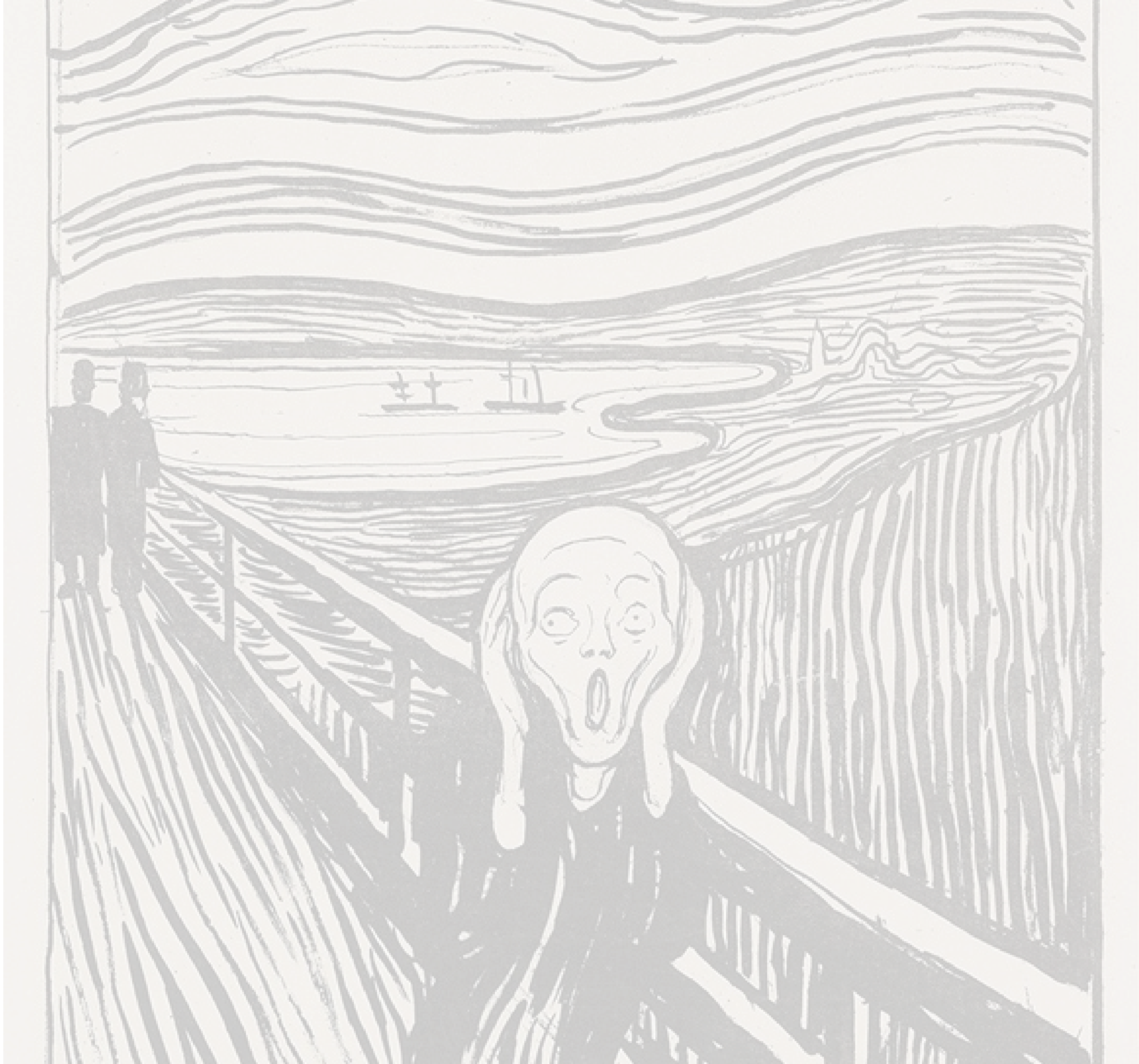
Idle Time

> Close the lobby early - show in progress



Late Event Handling

> What happens when an event shows up **after** the window is closed?



Dead Letter Queue

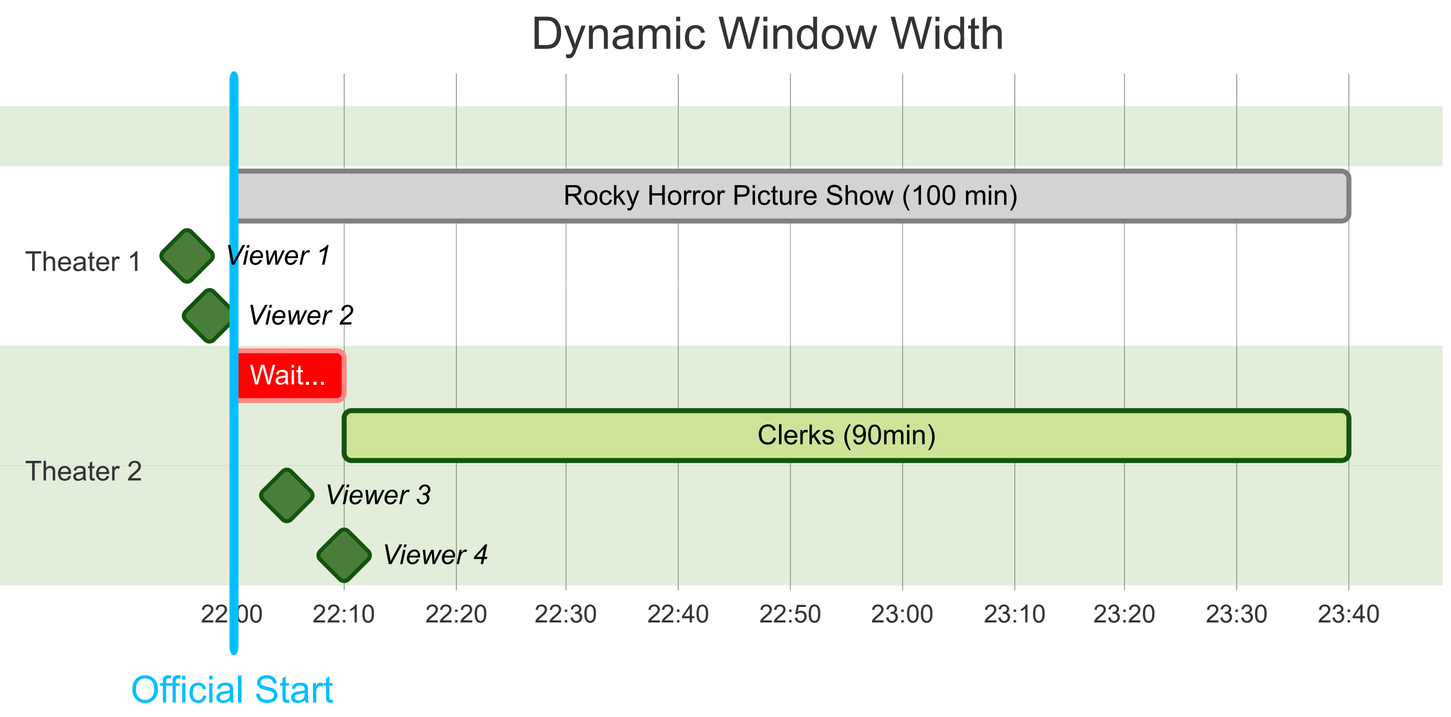
What ends up in DLQ?

- Malformed
  - $validate rejections
  - Payload deserialization errors
- Time Boundary Violations (late/early)
- Aggregation pipeline stage errors
- Full Document not available (change stream)

DLQ - How?

```
const options = {
  dlq: {
    connectionName: "my_dlq",
    db: "my_db",
    coll: "events_for_review"
  }
}

sp.createStreamProcessor("mySP", /** pipeline */, options);
```



Session Window

- Closes when no event seen `gap` time after latest.

Thank You

MongoDB Champions

Nuri Halperin

LinkedIn: @nurih

nuri@plusnconsulting.com