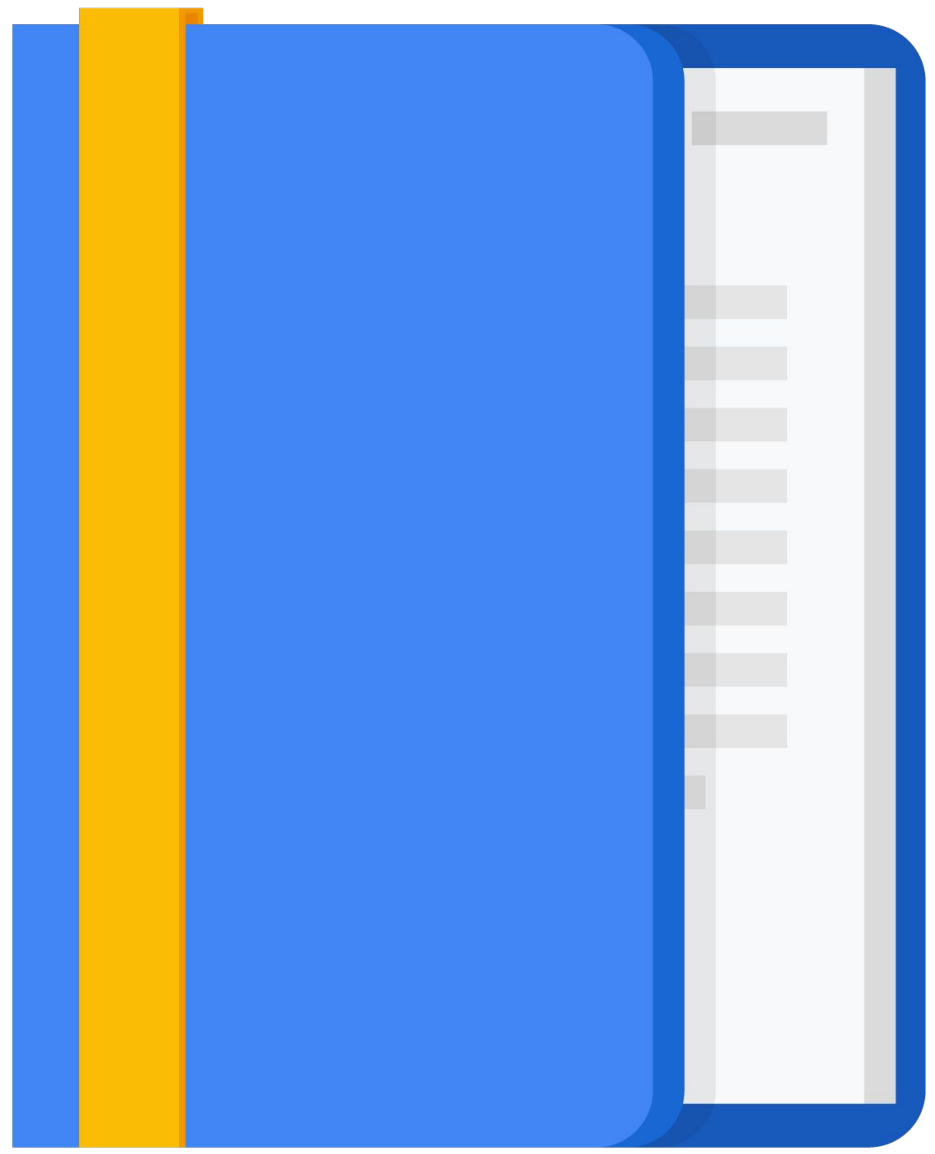
Windows, Watermarks, and Triggers

Israel Herraiz

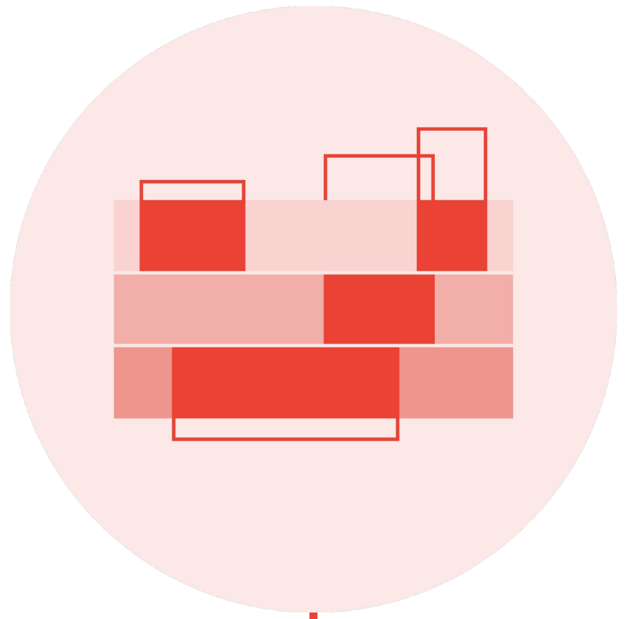
Strategic Cloud Engineer,
Google Cloud

Agenda



Developing pipelines

Agenda

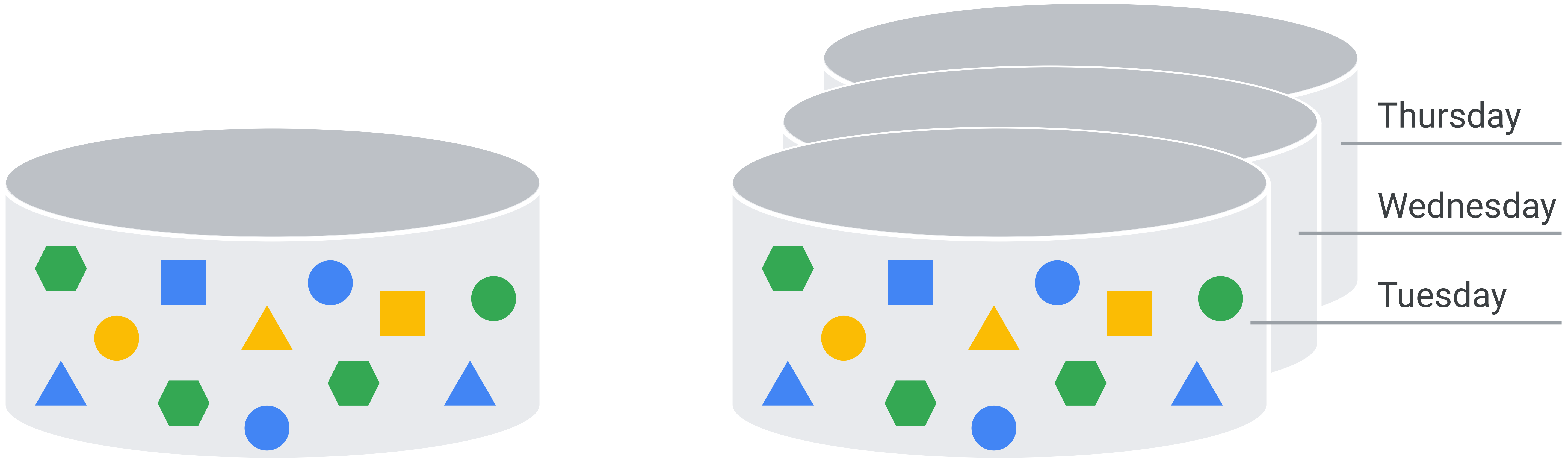


Windows

Watermarks

Triggers

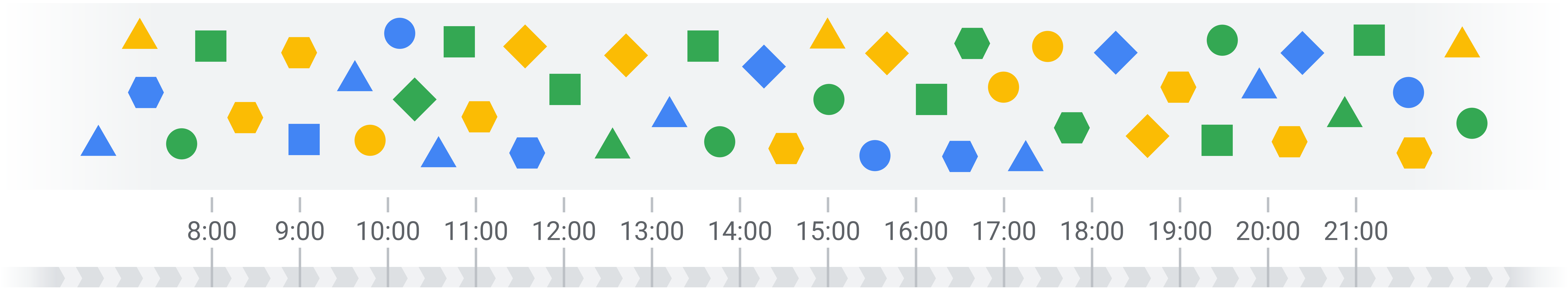
Batch vs. streaming



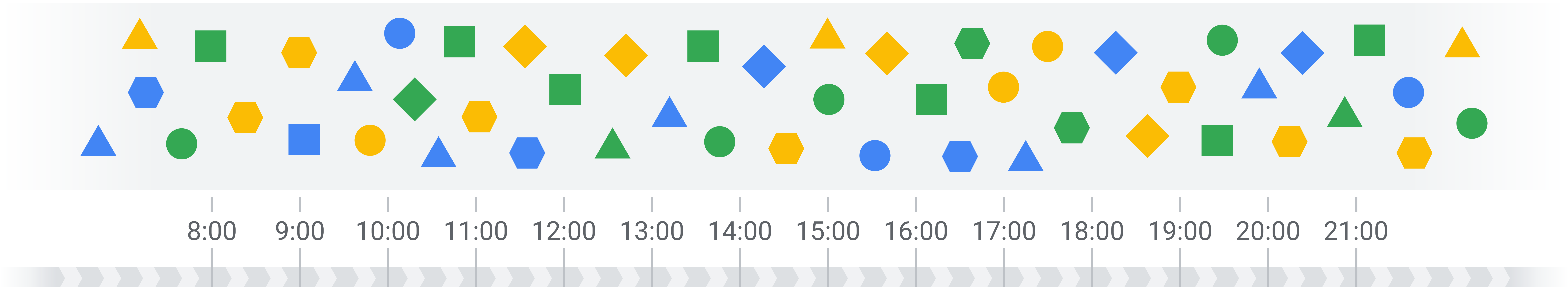
Low GB's

TB's to PB's

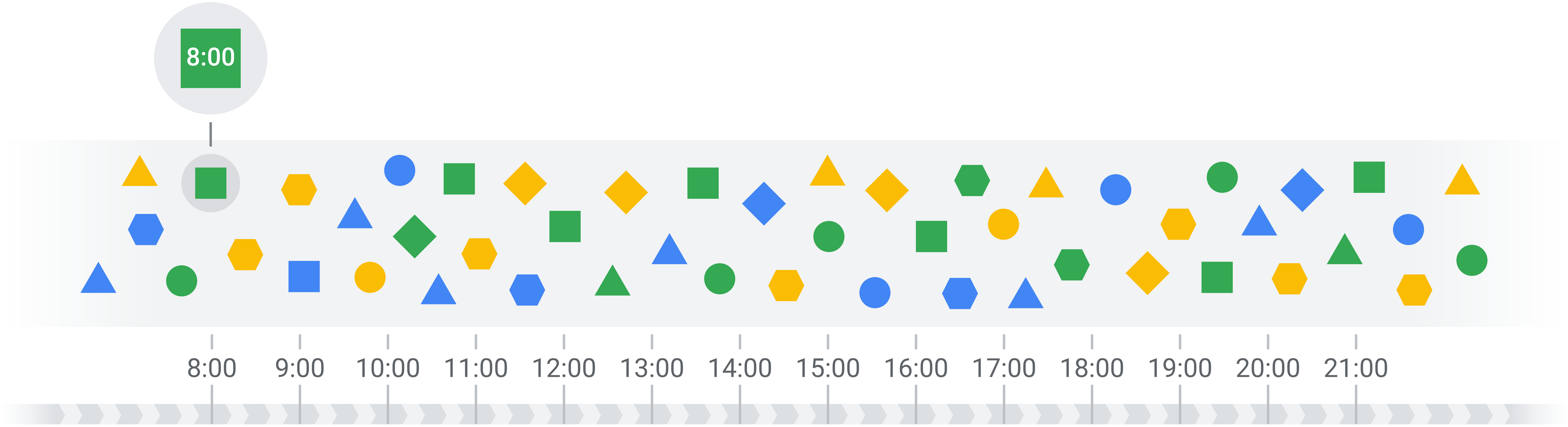
Data is not always stationary



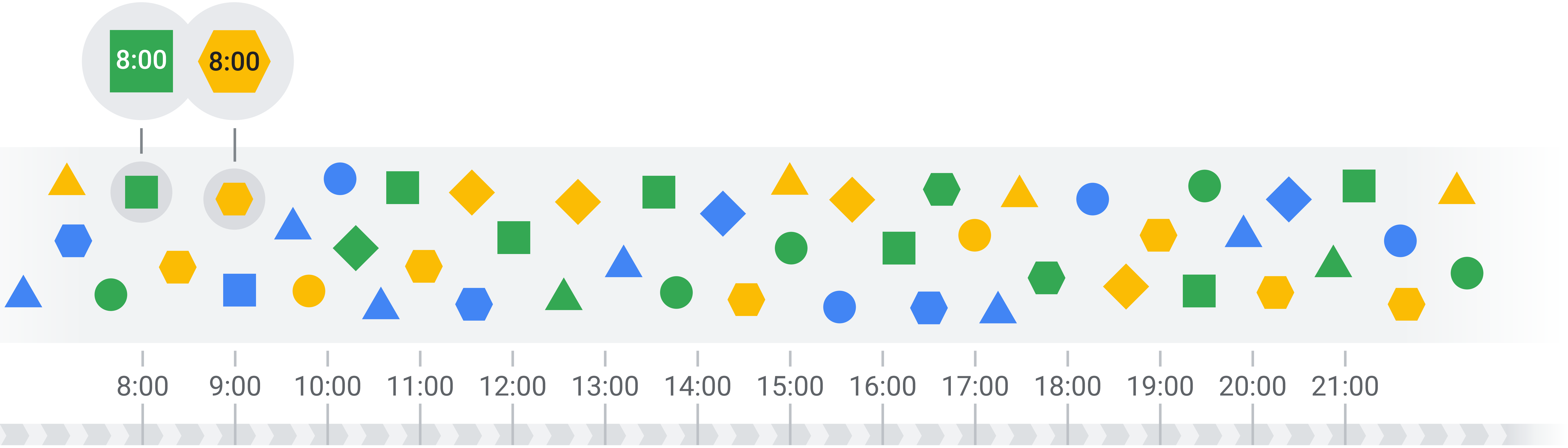
Out of order in streams



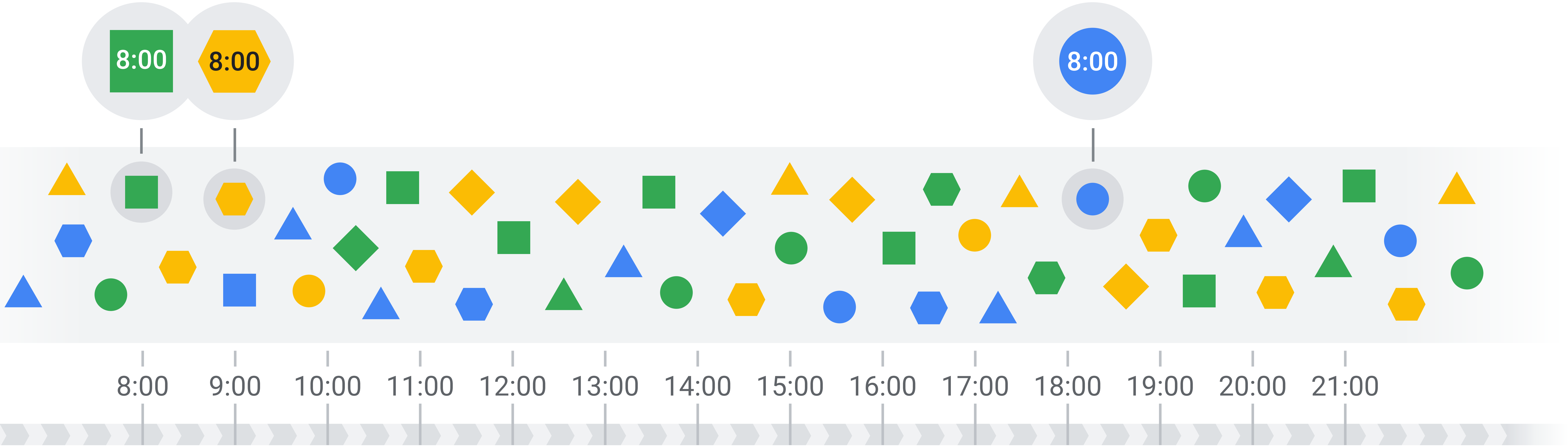
Out of order in streams



Out of order in streams

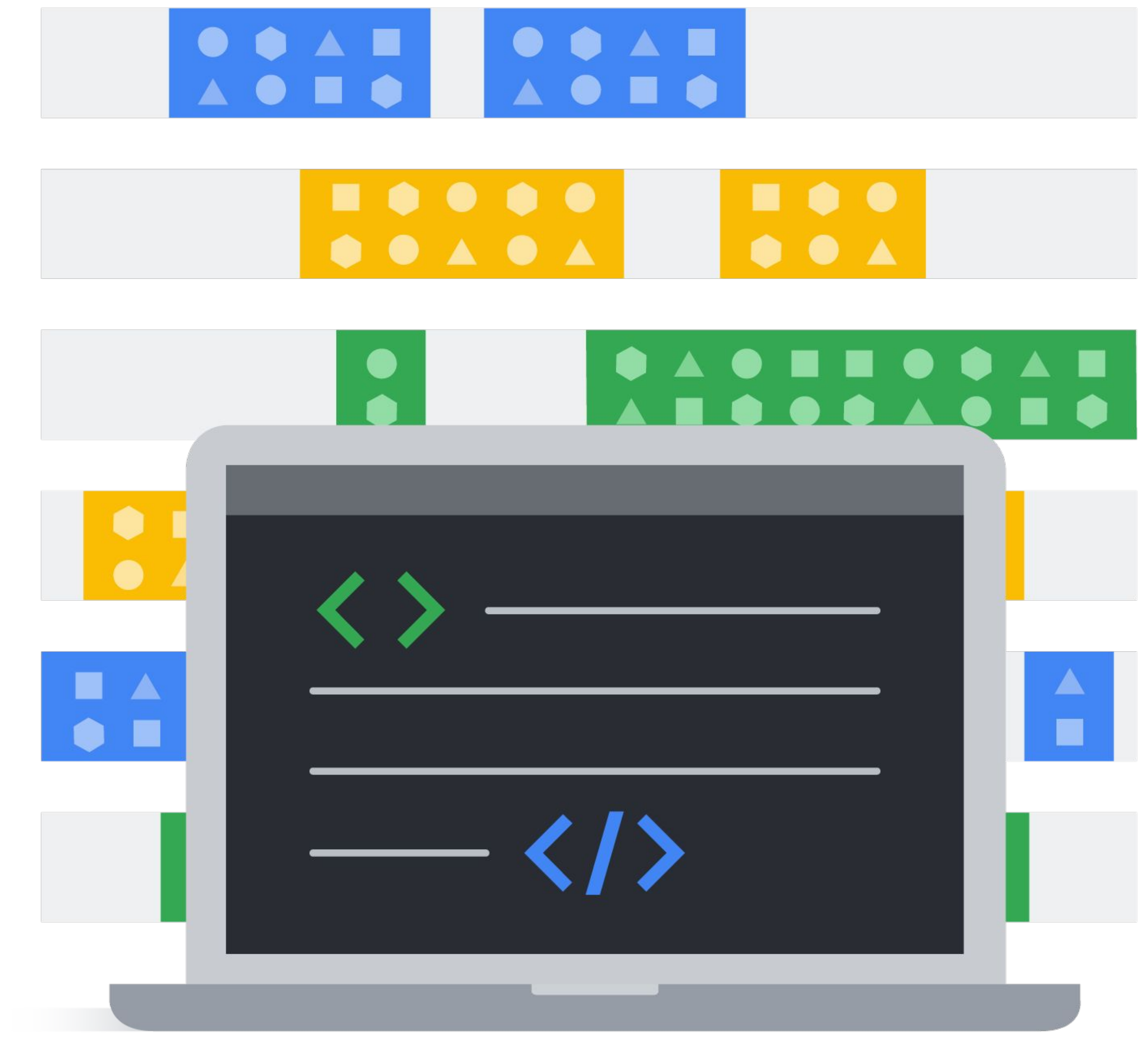


Out of order in streams



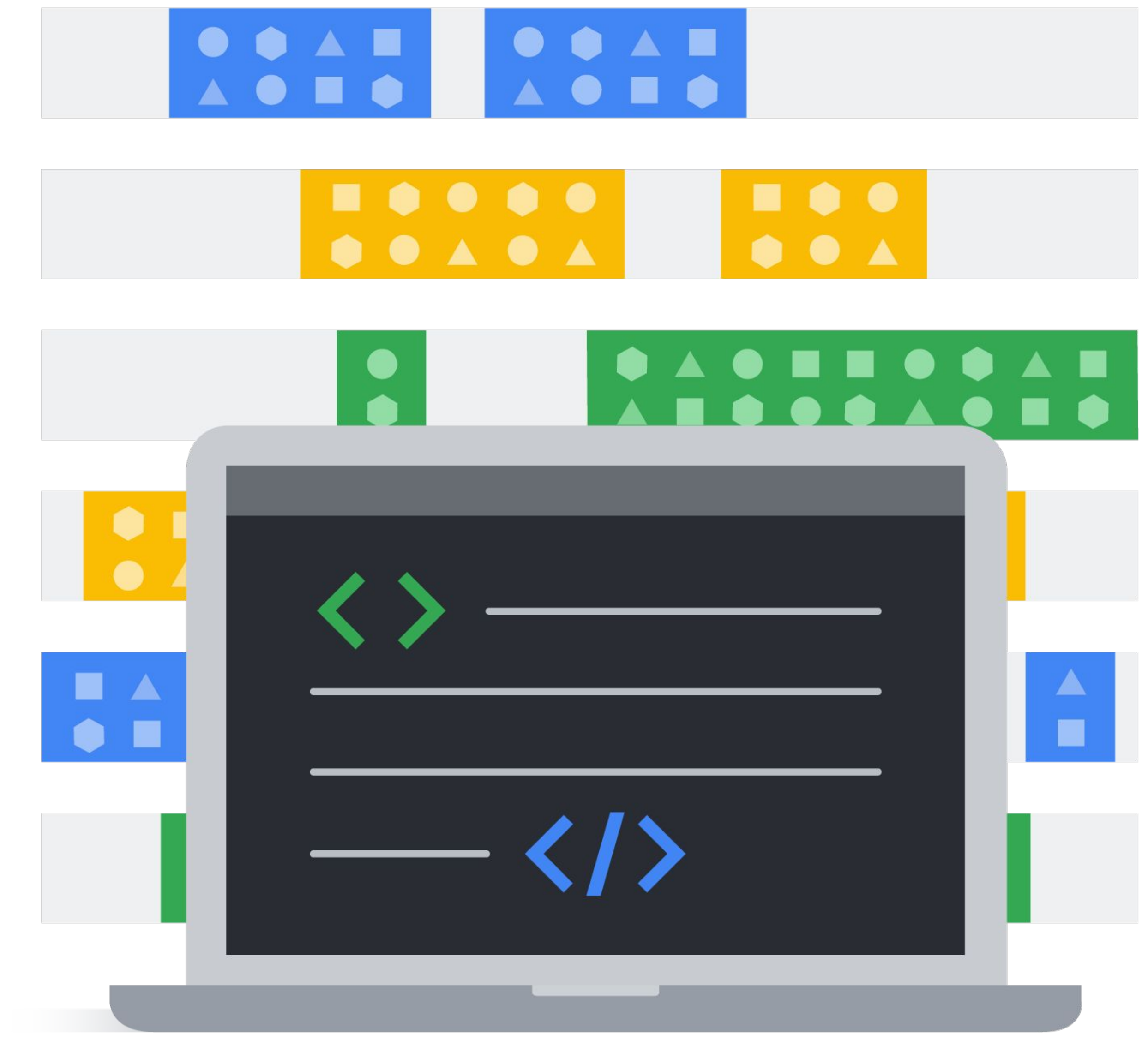
Windows

- Windowing divides data into time-based, finite chunks

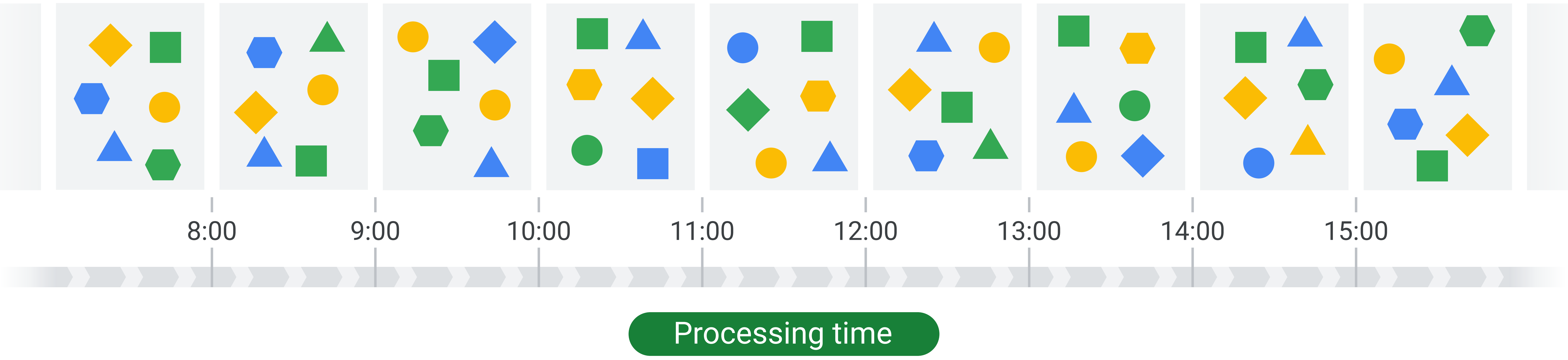


Windows

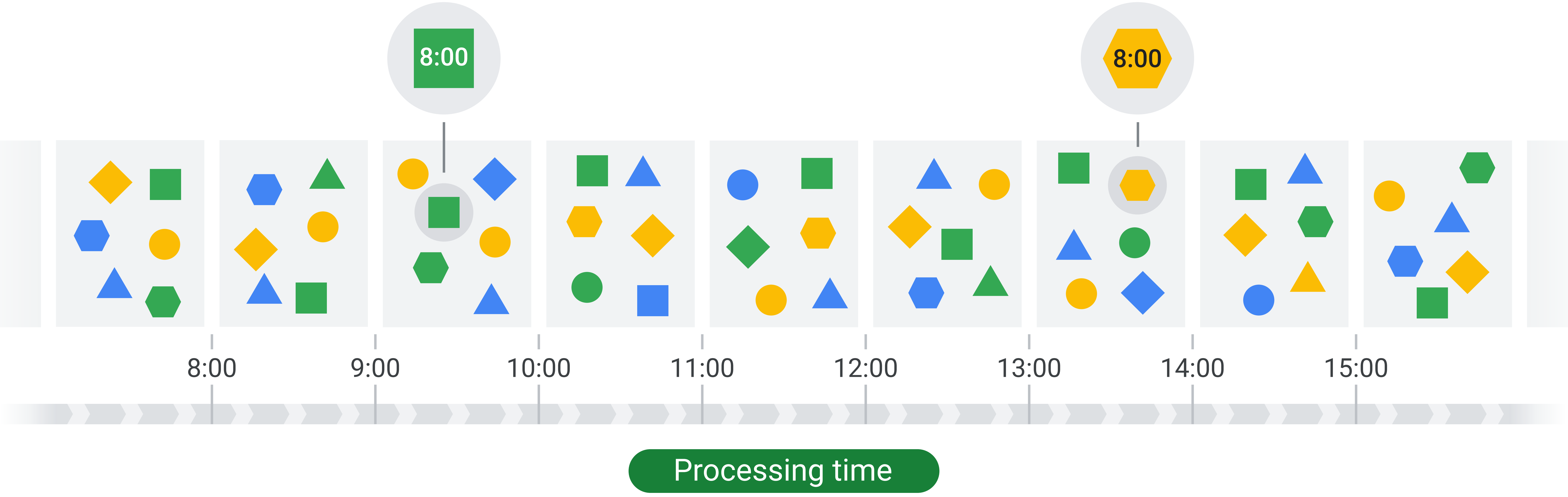
- Windowing divides data into time-based, finite chunks
- Required when doing aggregations over unbounded data using Beam primitives (GroupByKey, Combiners)
    - You can also do aggregations using state and timers



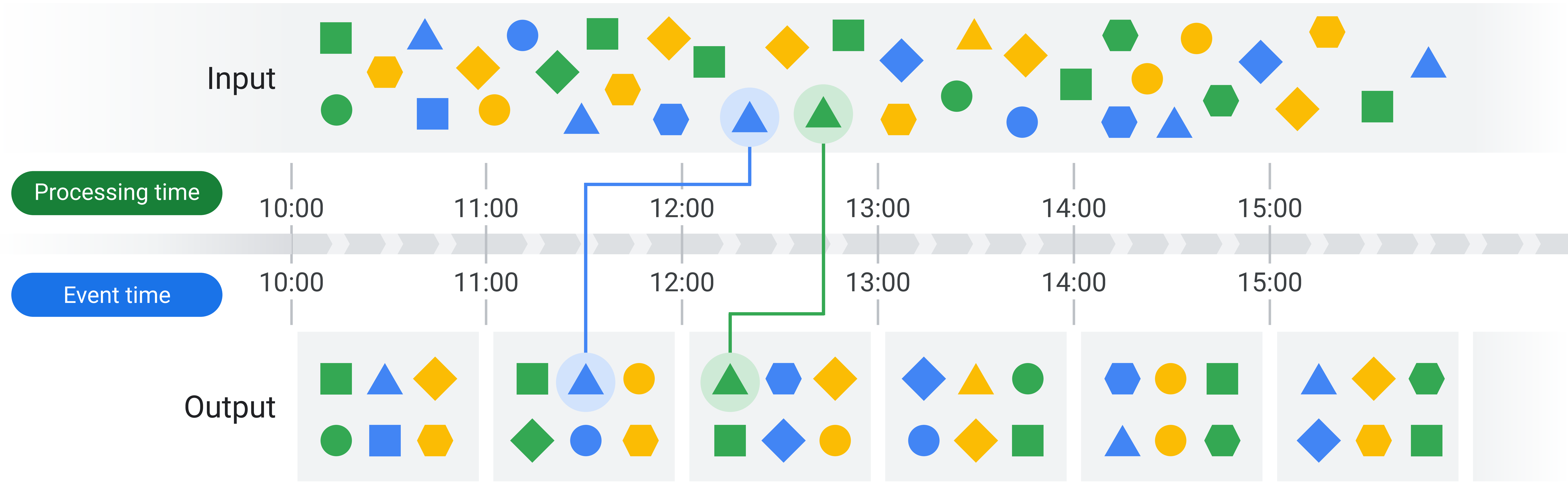
Windowing by processing time



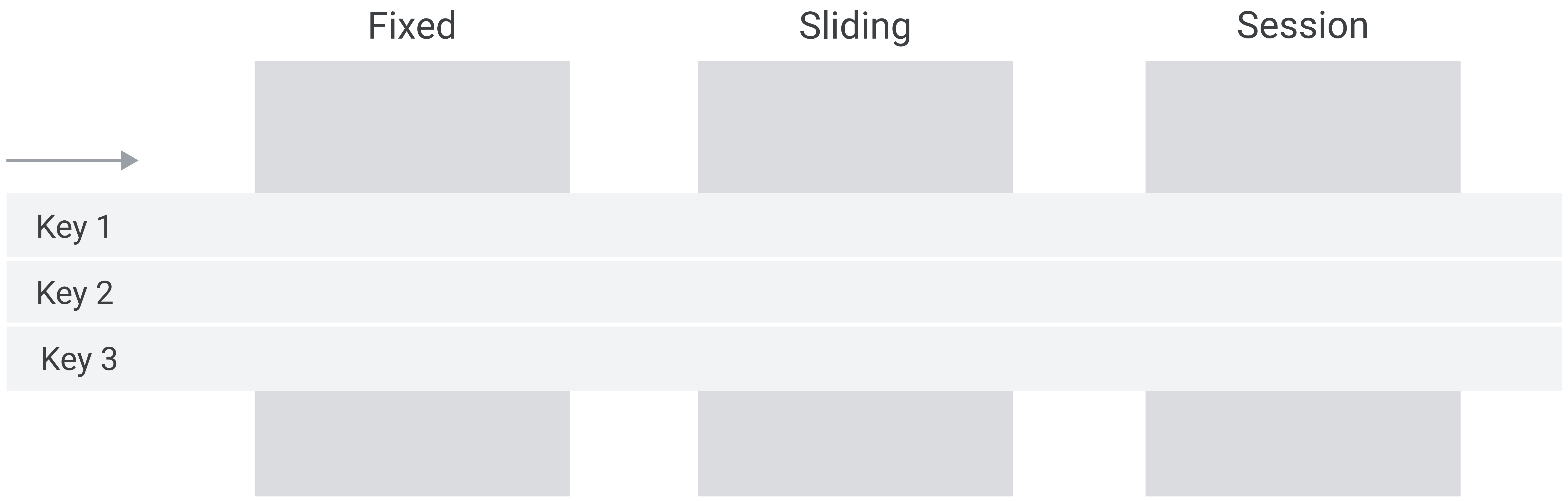
Windowing by processing time



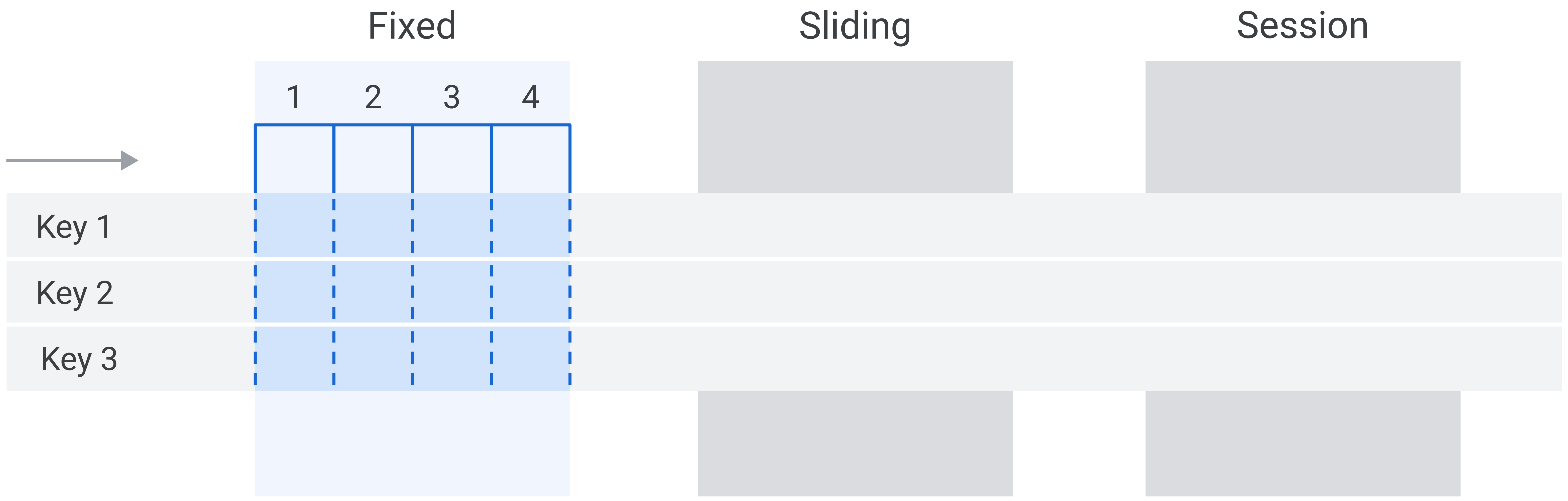
Windowing by event time



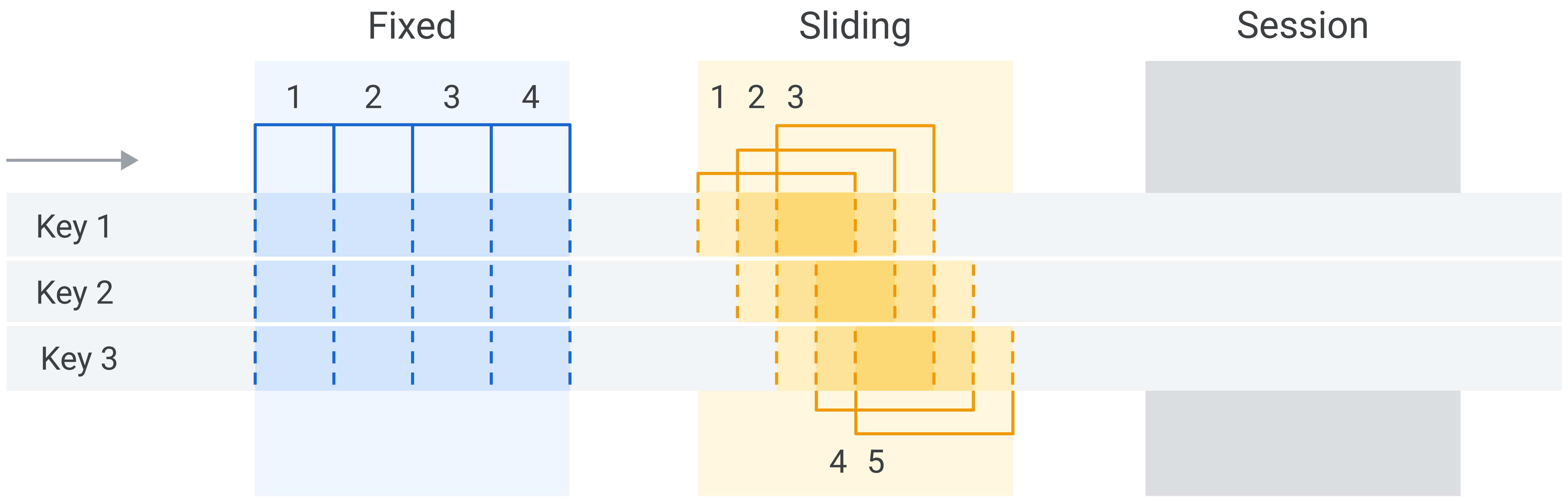
Types of windows in Apache Beam



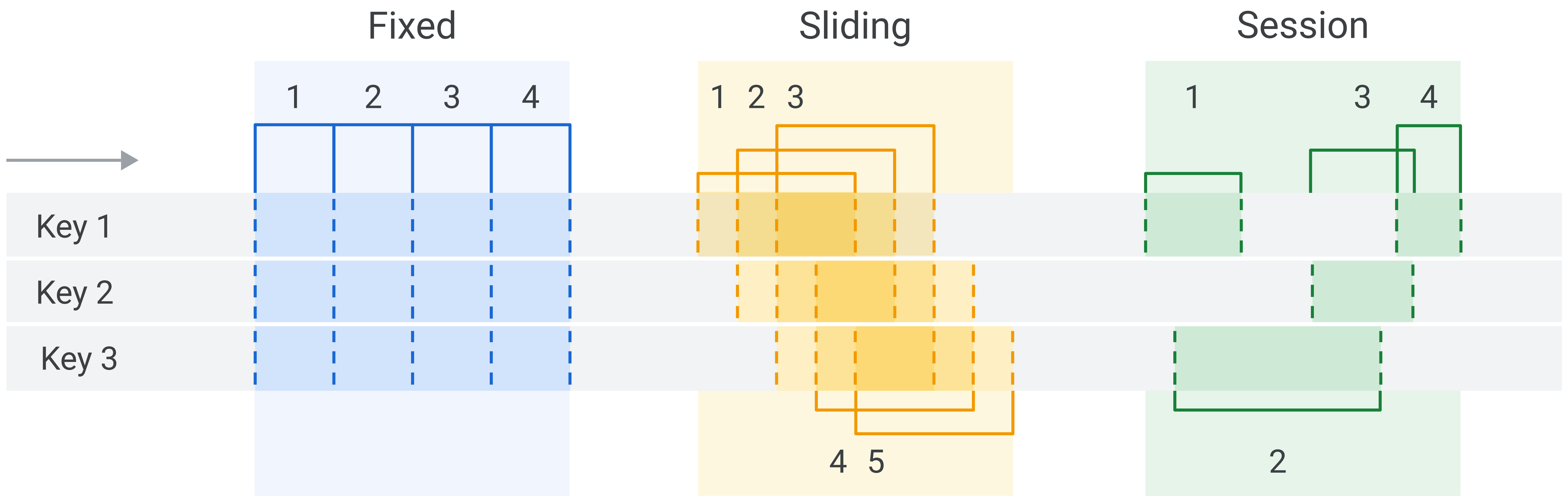
Types of windows in Apache Beam



Types of windows in Apache Beam



Types of windows in Apache Beam

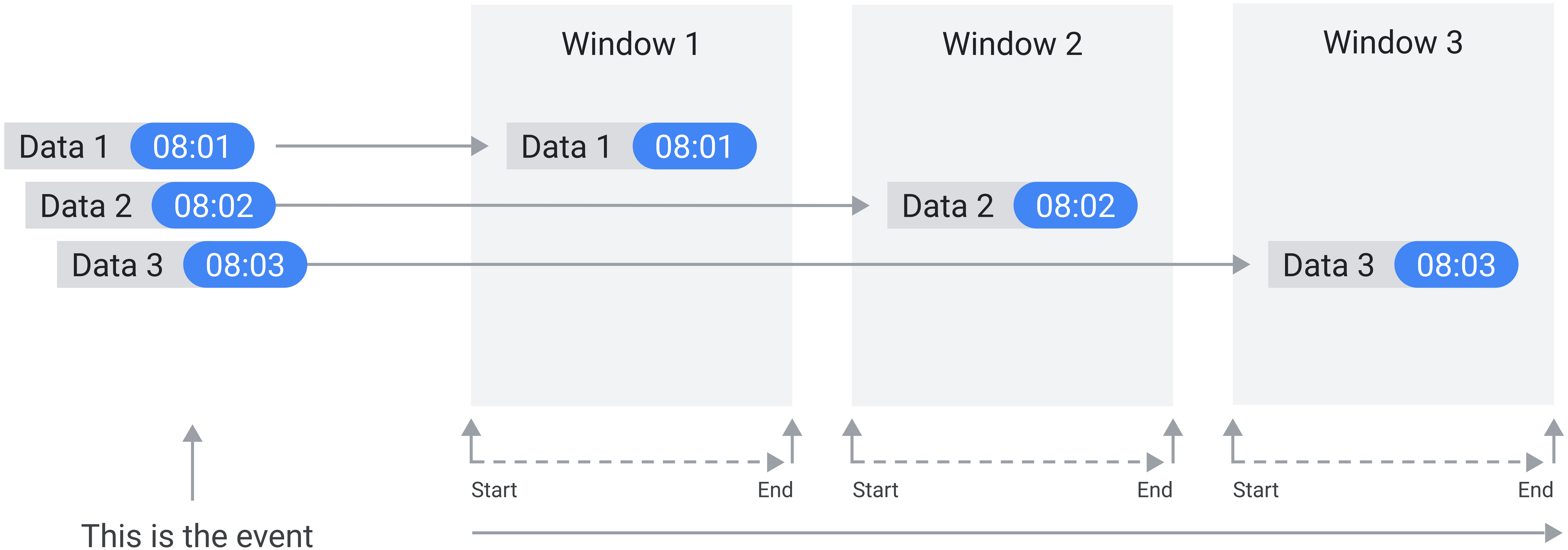


Developing pipelines

Agenda

- Windows
- Watermarks
- Triggers

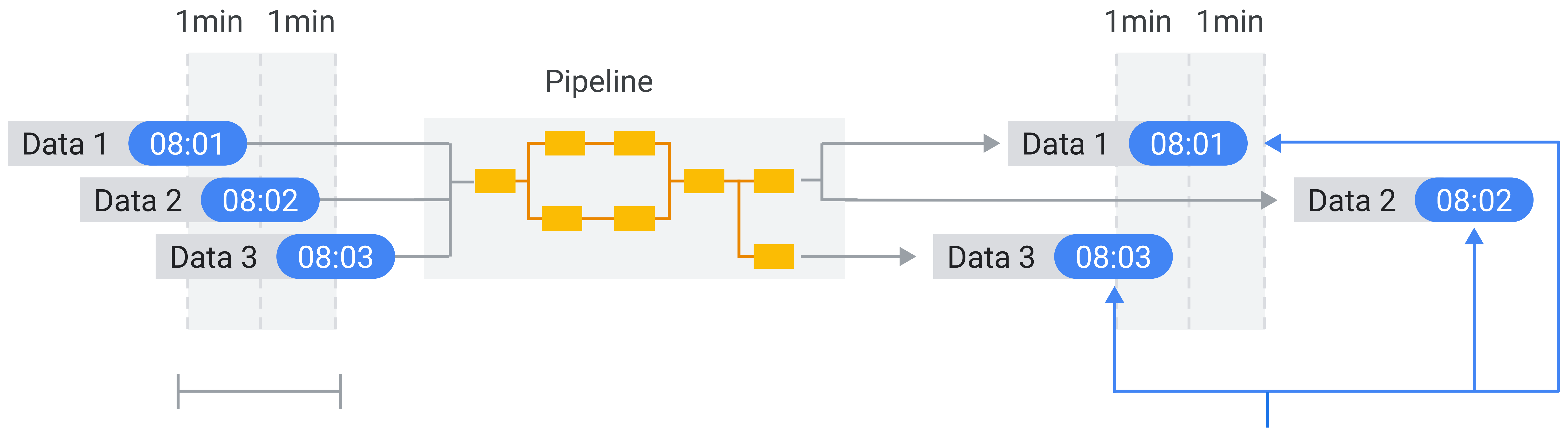
How does windowing work?



This is the event time when the event originated

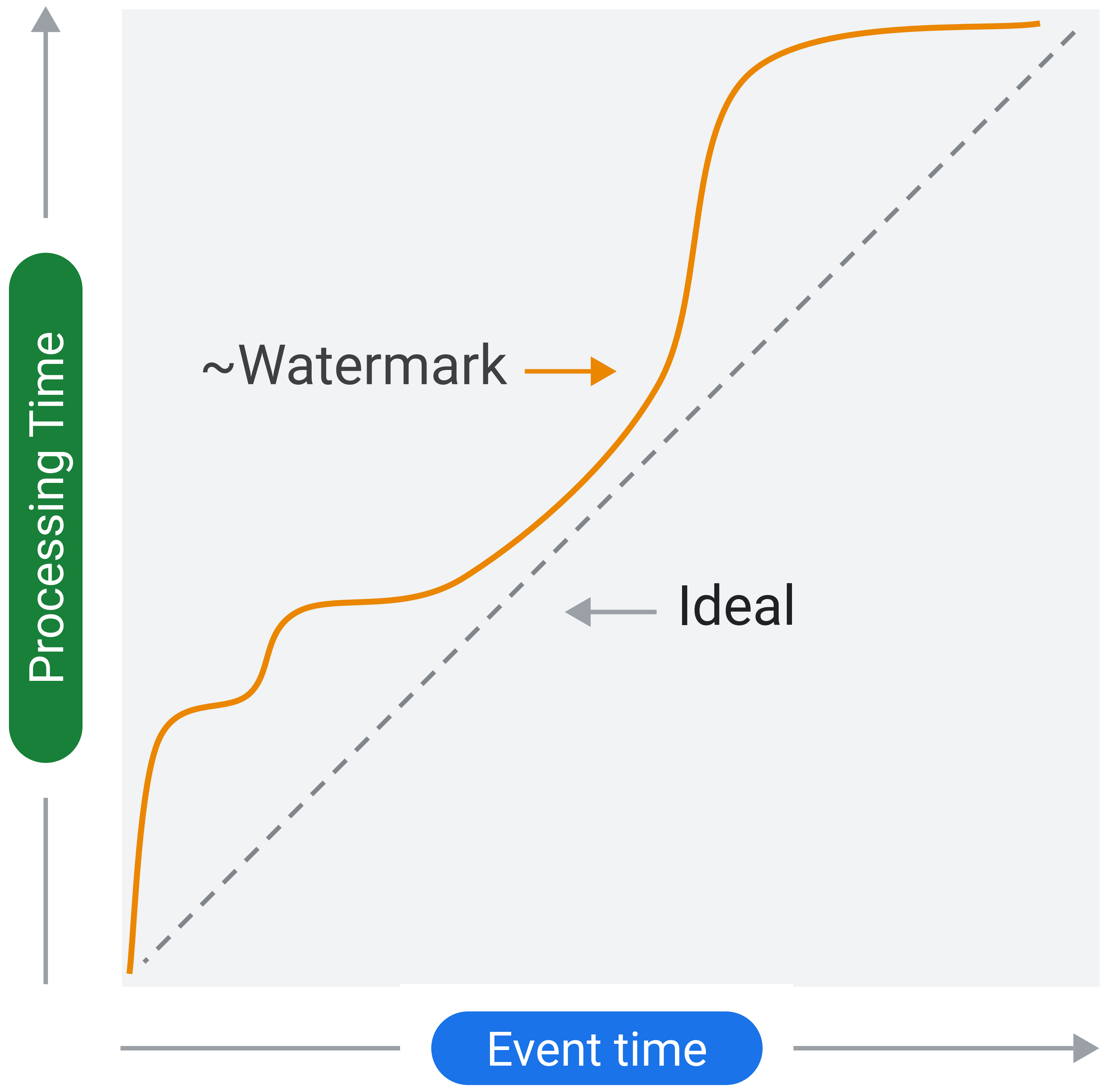
These windows are defined to be one minute long

Latency problem: when do we close the window?



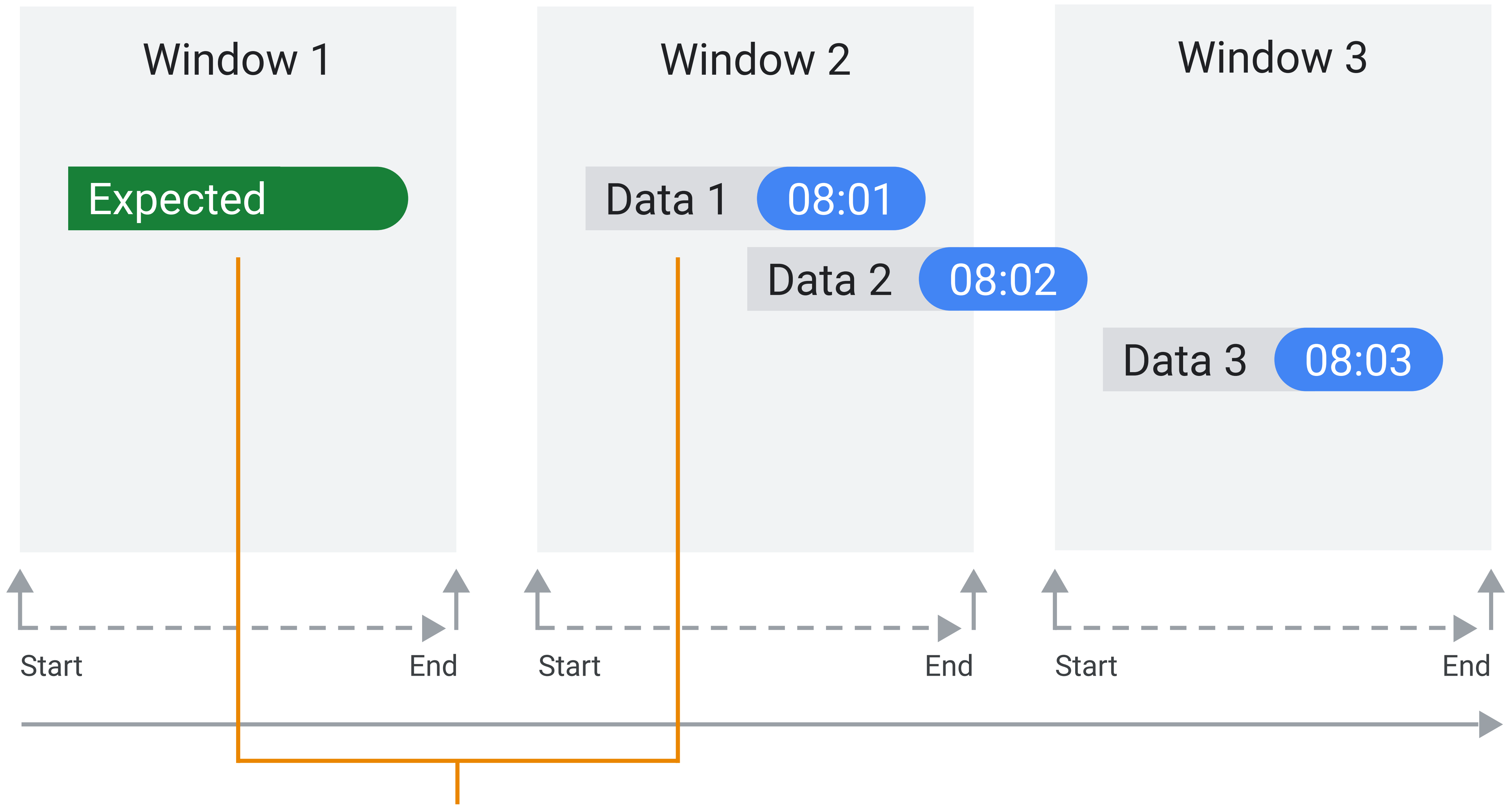
This is the event time when the event originated

The time relationships at origin are not preserved. They are arriving later than anticipated. And some of them are outside the original one-minute window.


Processing Time
~Watermark
Ideal
Event time

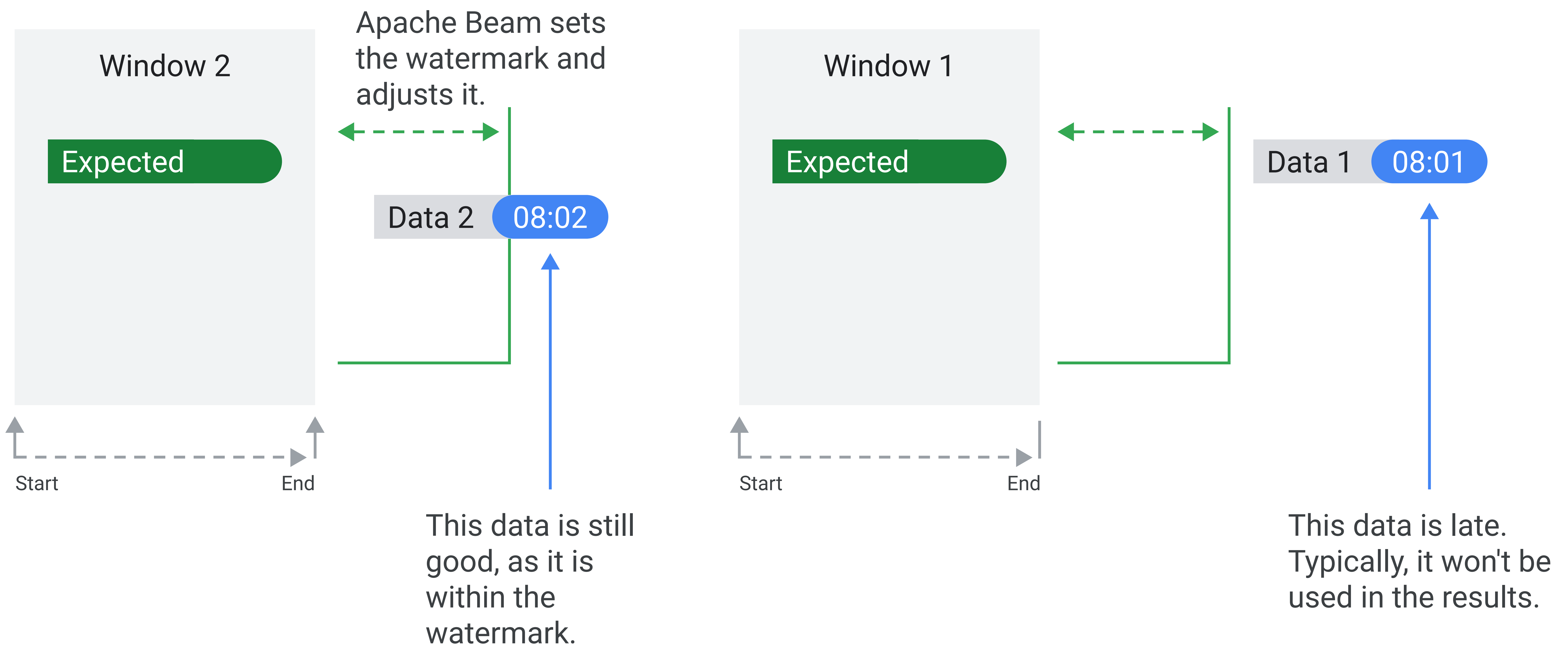
Introducing the watermark

The data could be a little past the window or a lot. Data 2 is a little outside of Window 2. Data 1 is completely outside of Window 1.

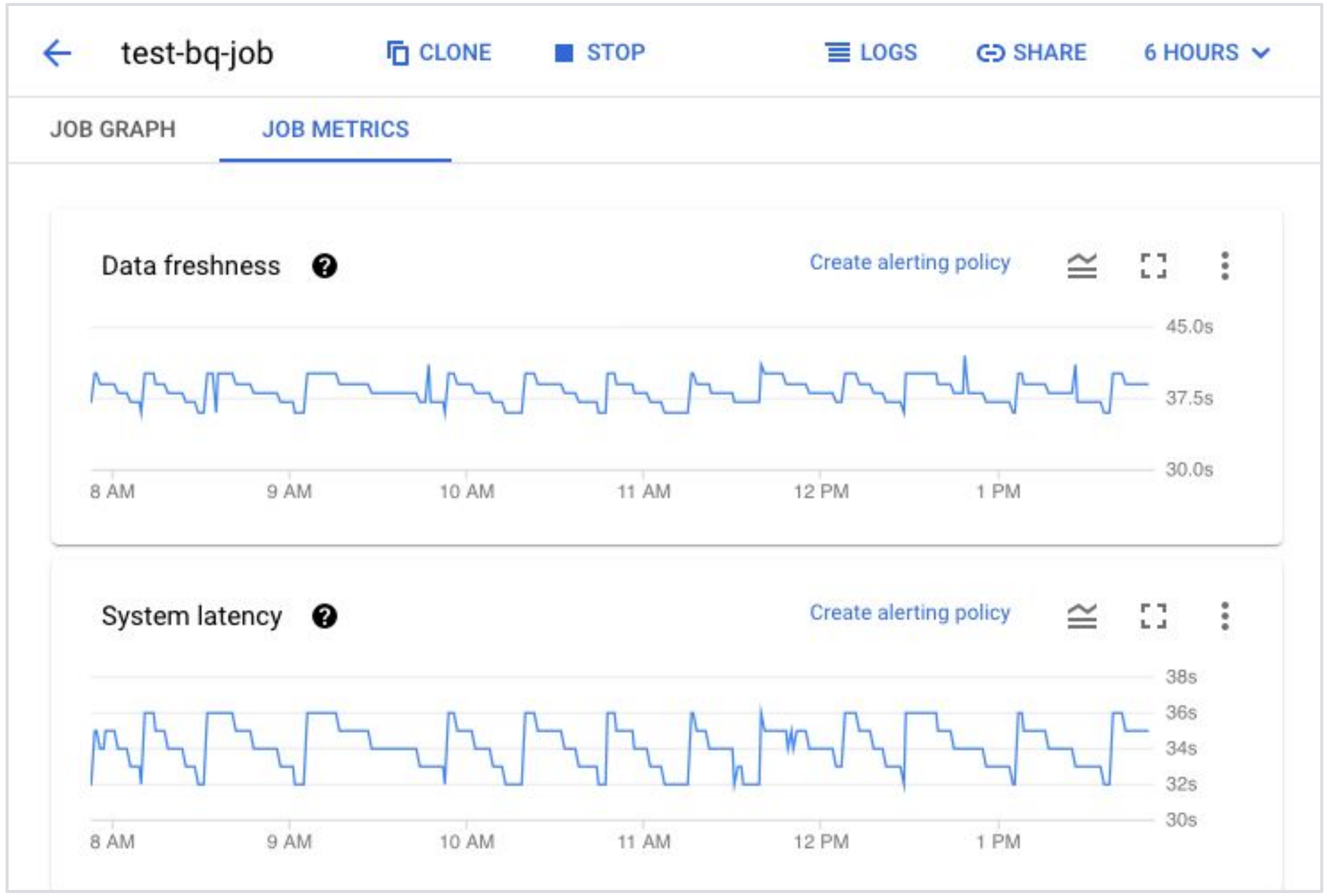


The difference in time from when data was expected to arrive and when it actually arrived is called the **lag time**.

Data is late in comparison to the watermark


Window 2
Expected
Apache Beam sets the watermark and adjusts it.
Data 2
08:02
Start
End
This data is still good, as it is within the watermark.
Window 1
Expected
Data 1
08:01
Start
End
This data is late. Typically, it won't be used in the results.

How do you observe the watermark in Dataflow?



Data freshness
The amount of time between real time and the output watermark.

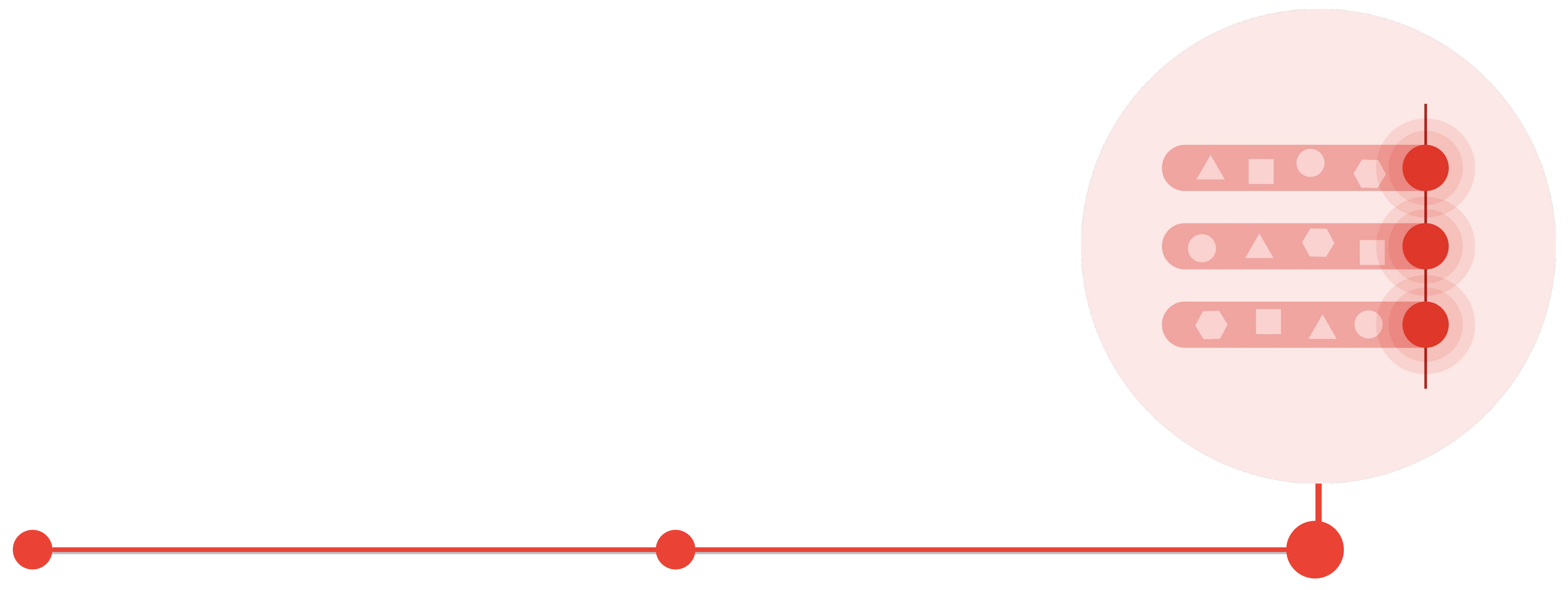
System latency
System latency is the current maximum duration that an item of data has been processing or awaiting processing.

Data freshness and system latency

	Stable data freshness	Ever-increasing data freshness
Stable system latency	**Ideal** ✓ Pipeline processing data at good pace.	**Monitor** ⊖ Data is accumulating at the input. More workers are needed. Autoscaling will spin up new workers (backlog size).
Ever-increasing system latency	**Monitor** ⊖ Complex processing, messages take longer to be processed. Autoscaling likely to spin up new workers (high CPU usage)	**Risk** ⚠ Complex processing and data is accumulating at the input. Autoscaling will spin up new workers (backlog size and CPU usage).

Developing pipelines

Agenda

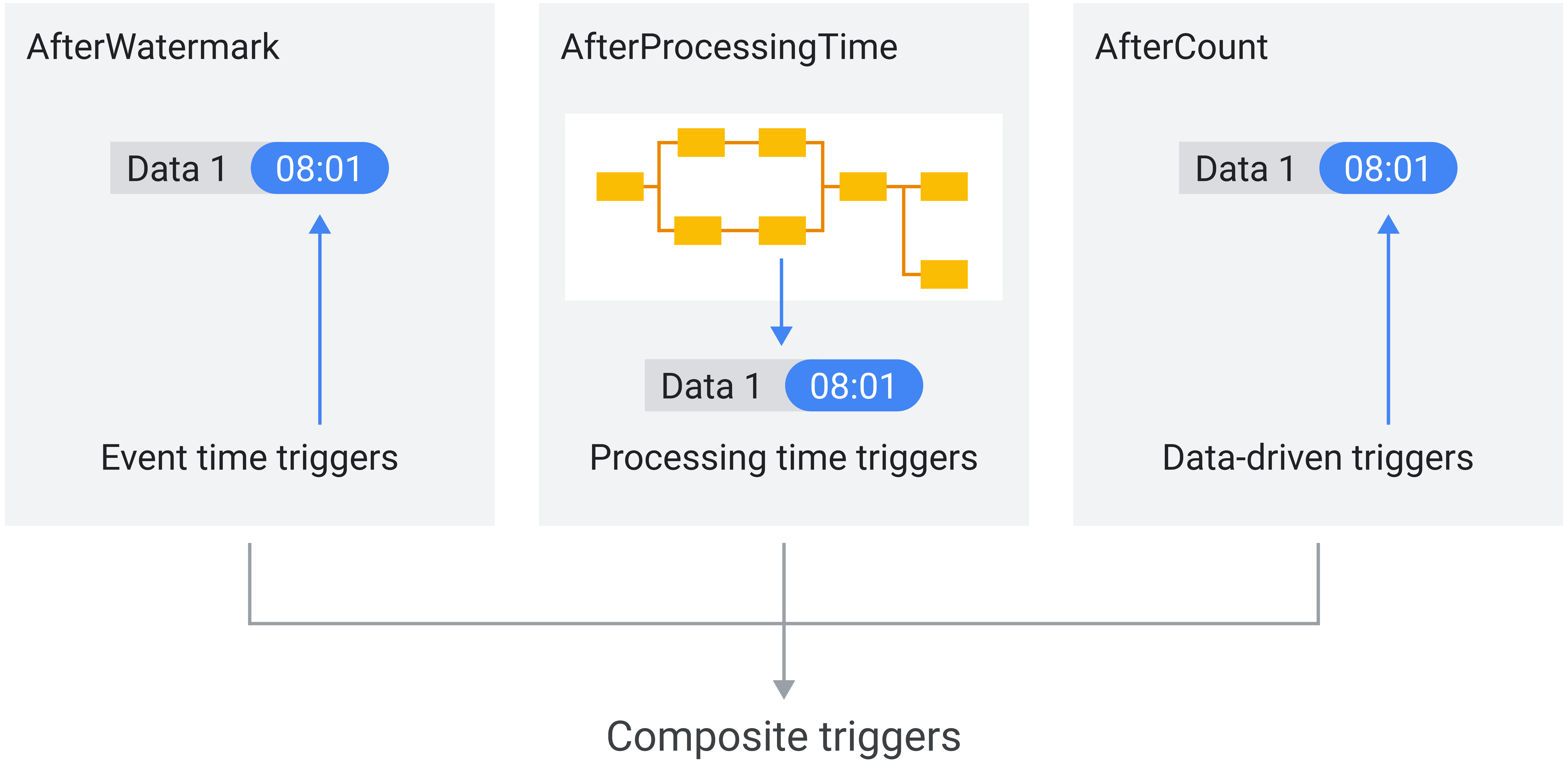


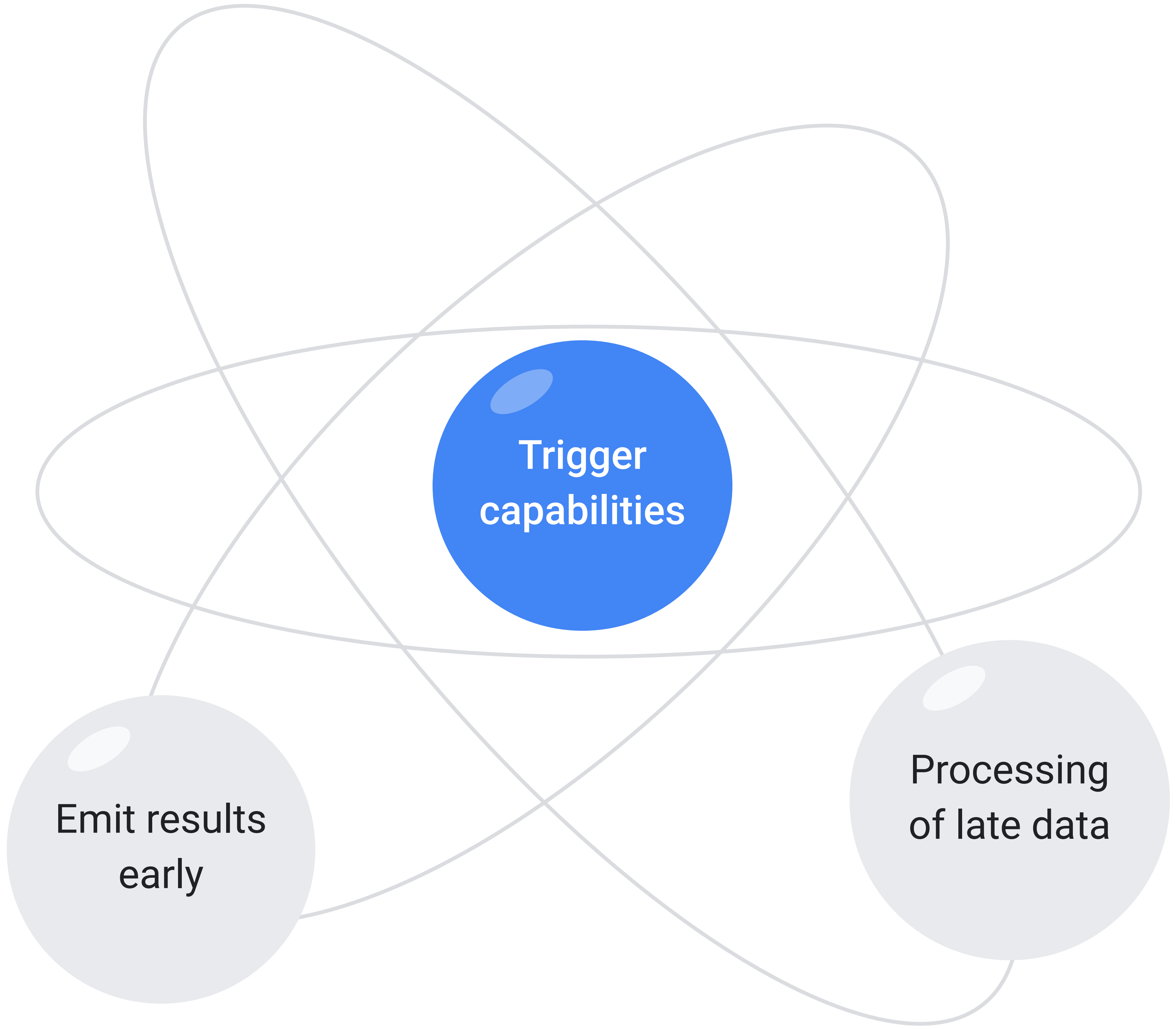
Triggers

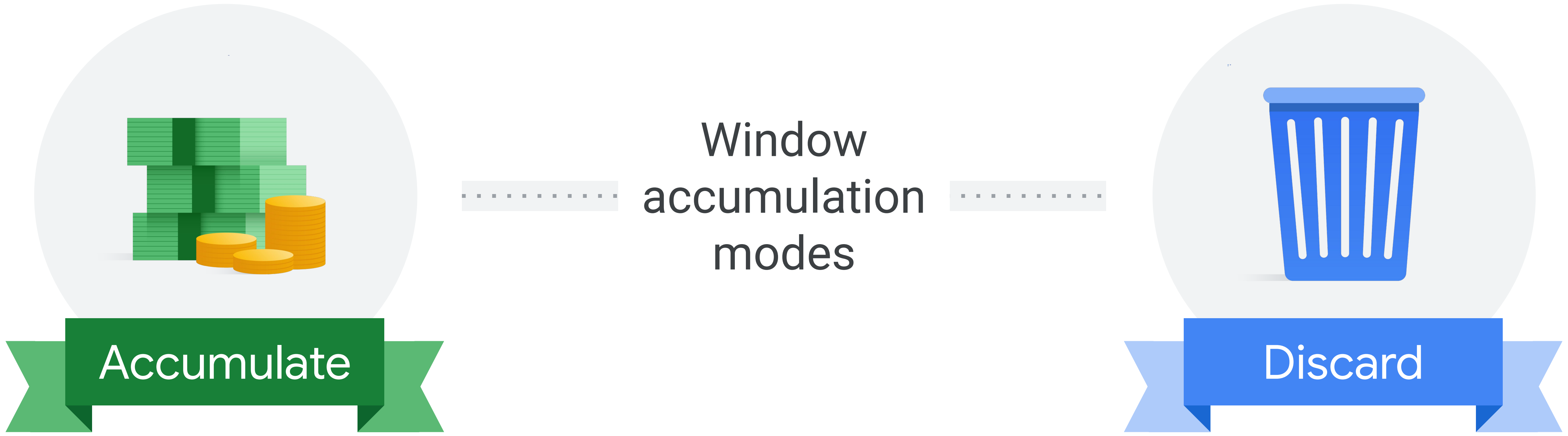
Triggers

Event time	They operate on event time
Processing time	They operate on processing time
Composite	They combine multiple triggers
Data-driven	They operate by examining the data as it arrives and firing when it meets a certain condition (*) Note: *Only support firing after a certain number of elements*

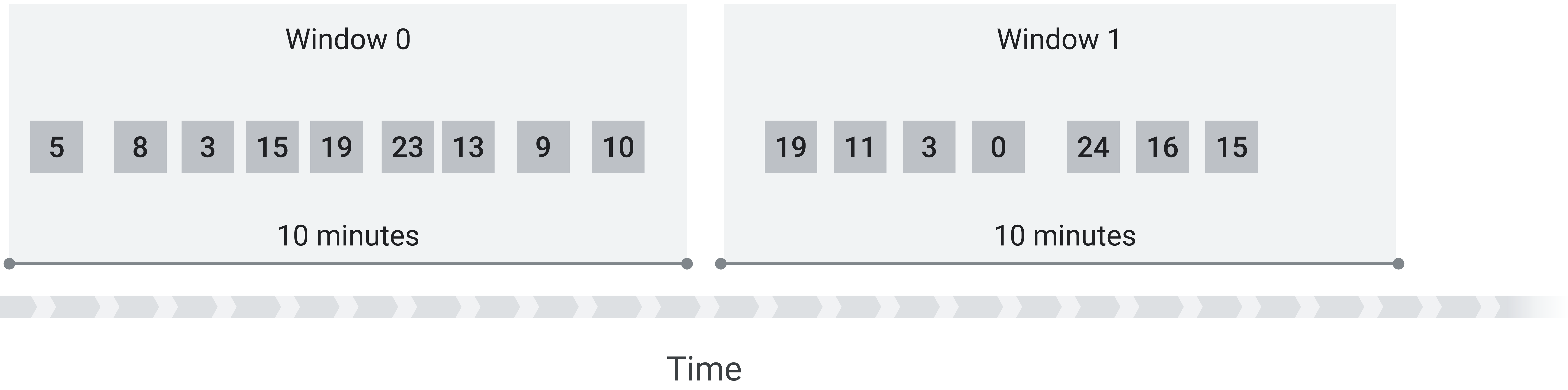
Custom triggers




Trigger capabilities
Emit results early
Processing of late data

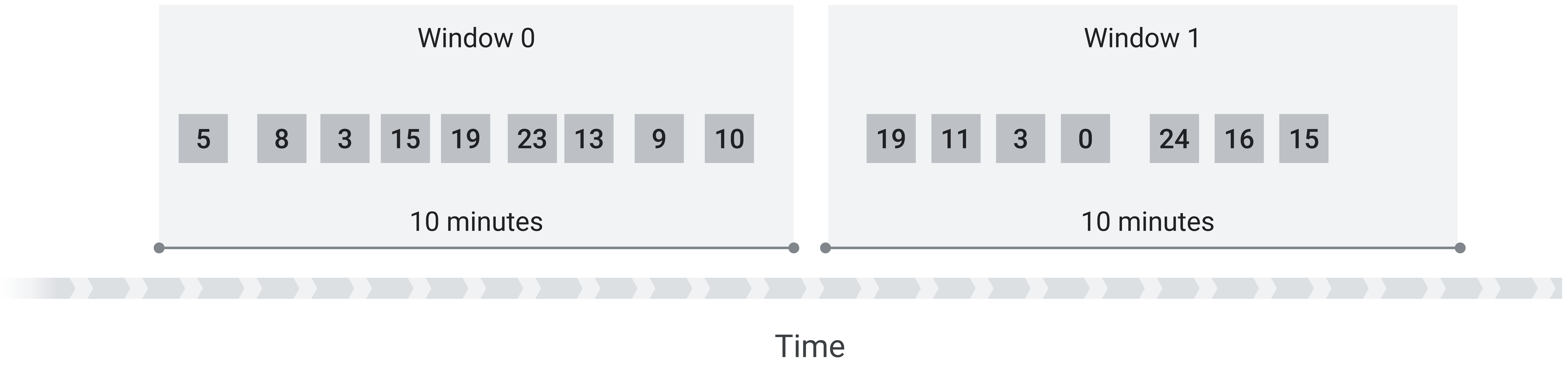

Accumulate
Window accumulation modes
Discard

Accumulate



1st	trigger firing	[5, 8]
2nd	trigger firing	[5, 8, 3, 15, 19, 23]
3rd	trigger firing	[5, 8, 3, 15, 19, 23, 9, 13, 10]

Discard



1st	trigger firing	[5, 8]
2nd	trigger firing	[3, 15, 19, 23]
3rd	trigger firing	[9, 13, 10]

Python: Code examples

```
pcollection | WindowInto(
    SlidingWindows(60, 5),                                        # Sliding window of 60 seconds, every 5 seconds
    trigger=AfterWatermark(                                       # Relative to the watermark, trigger:
      early=AfterProcessingTime(delay=30),                        # -- fires 30 seconds after pipeline commences
      late=AfterCount(1))                                         # -- and for every late record (< allowedLaten4ess)
    accumulation_mode=AccumulationMode.ACCUMULATING,              # the pane should have all the records
    allowed_lateness=Duration(seconds=2*24*60*60))                # 2 days
```

```
pcollection | WindowInto(
    FixedWindows(60),                                             # Fixed window of 60 seconds
    trigger=Repeatedly(                                           # Set up a composite trigger that triggers
        AfterAny(                                                 # whenever either of these happens:
            AfterCount(100),                                      # -- 100 elements accumulate
            AfterProcessingTime(1 * 60))),                        # -- every 60 seconds (ignore watermark)
    accumulation_mode=AccumulationMode.DISCARDING,                # the trigger should be with only new records
    allowed_lateness=Duration(seconds=2*24*60*60))                # 2 days
```

Java: Code examples

```
pcollection.apply(
        Window.<String>into(
                SlidingWindows.of(Duration.standardSeconds(60)).every(Duration.standardSeconds(5)))
            .triggering(
                AfterWatermark.pastEndOfWindow()
                    .withEarlyFirings(
                        AfterProcessingTime.pastFirstElementInPane()
                            .plusDelayOf(Duration.standardSeconds(30)))
                    .withLateFirings(AfterPane.elementCountAtLeast(1)))
            .discardingFiredPanes()
            .accumulatingFiredPanes().withAllowedLateness(Duration.standardDays(2)));
```

```
pcollection.apply(
        Window.<String>into(FixedWindows.of(Duration.standardSeconds(60)))
            .triggering(
                Repeatedly.forever(
                    AfterFirst.of(
                        AfterPane.elementCountAtLeast(100),
                        AfterProcessingTime.pastFirstElementInPane()
                            .alignedTo(Duration.standardSeconds(60)))))
            .discardingFiredPanes()
            .withAllowedLateness(Duration.standardDays(2)));
```