EVIDEN

03 Data Pipeline 2

Event processing & Model serving

Event driven architecture

Context & challenges

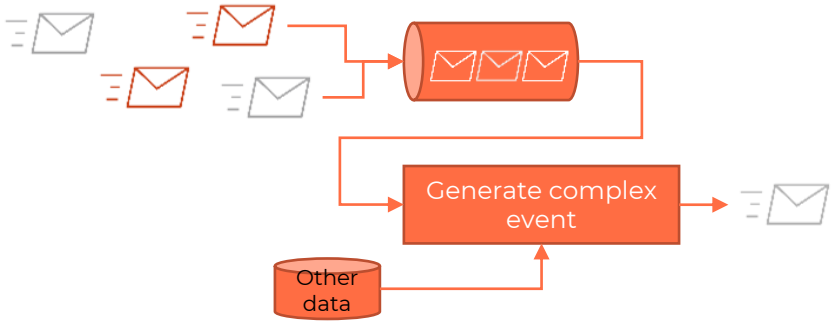
Event : something that happened somewhere ;D (frontend, IoT, API, ...)

Availability : events can be created anytime, so streaming engine should always be up, ready and listening for messages

Event processing : extract valuable information from events

Performance : sometimes we have to deal with thousand of events/sec (volumetry) but most of the time the challenge is rather to process them as fast as possible (velocity) and expose results to the rest of the company

Complex event processing : combine multiple events to produce a complex information (customer logged in + shopping cart dropped => retention)



Coherence : Ensure business coherence over distributed system and messages is quite hard (heavily relay on tracing). At least, at most and exactly once mode are here for better consistency.

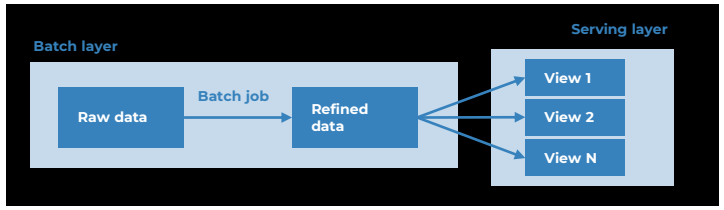
Late events : Since events pass through multiple layers before arriving in processing systems, order is not guaranteed. We have to wait for late events.

Event driven architecture

From traditional batch to streaming

What kind of architecture have you already used ?

Batch only

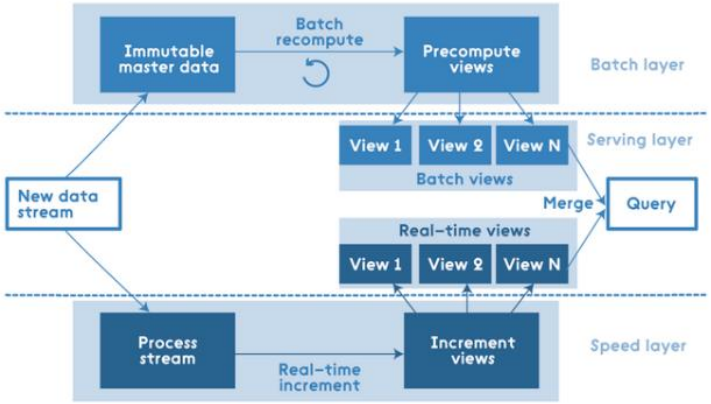


Processing done with huge **integration batches**, data is served for analytics purposes

Files, tables are easier to monitor/operate than unitary event (need cohérence checks, strong data lineage, etc)

Too slow to follow or anticipate rapid trends
Huge consumption spikes while idling the rest of the time

Lambda
Batch+Stream

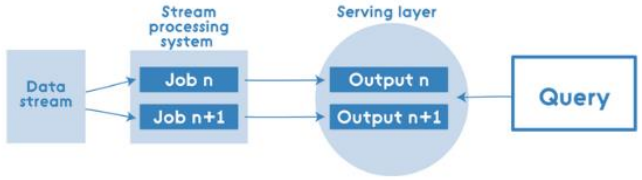


Same processing done by stream and batch layers, **reconciliation** is done in the serving layer

Reconciles batch and streaming
Keep a copy of the raw data
Allows data to be reprocessed

Maintain 2 different frameworks for the same output
Interactions with other bricks difficult because of the two paradigms

Kappa
Stream only



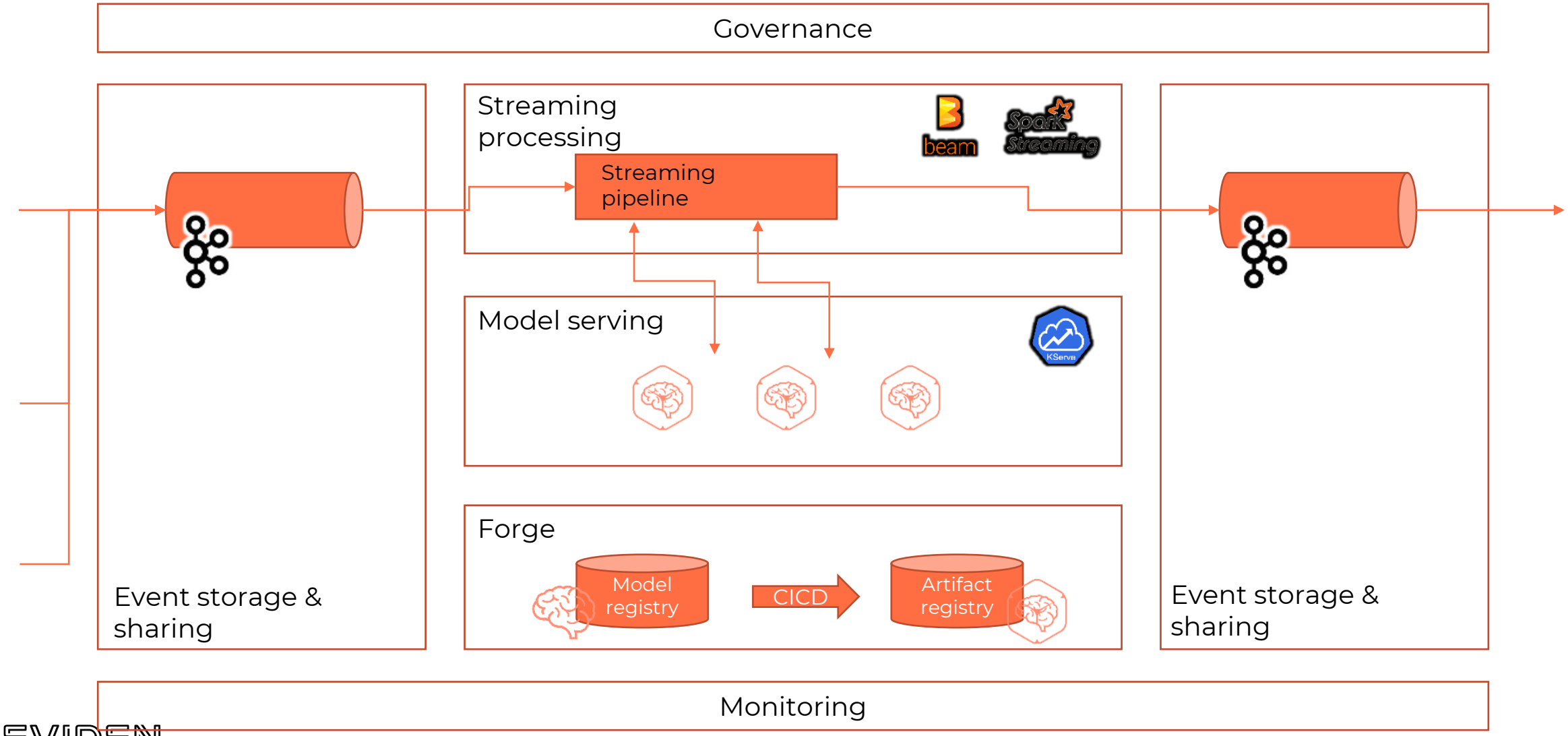
The idea is to use a **streaming sytem** that would retain **all the data**. At each evolution, we can reprocess old data to make them available with new version

1 single code to address the batch and the stream
Data reprocessing only when changing code
"Soft" activation of the new version (Canary deployment)

Need space to store all the data
Some data does not necessarily have to be transformed into events
It's not a "silver bullet" that solves all problems

Event driven architecture

Applied to ML



Streaming processing

The important concepts

- Time
  - Important distinction between Processing time and Event time will be the base for other more complex notions

- Windowing
  - Unbounded data stream need to be partitioned over time for effective data manipulation
  - Windowing is tightly linked with processing time

- Watermark
  - At every moment (processing time), this mark defines the limit in time until which the data has been observed (event time) by the system : no more data older (event time) than this mark will ever be seen again (if it is the case, the data is discarded)
  - Watermark is tightly linked with event time

- Trigger
  - Defines when an output should be persisted out of the observing windows (a window could have multiple triggers)

- Accumulation
  - When there's multiple triggers on a window, we need to decide what accumulation over all these value sin the window is applied

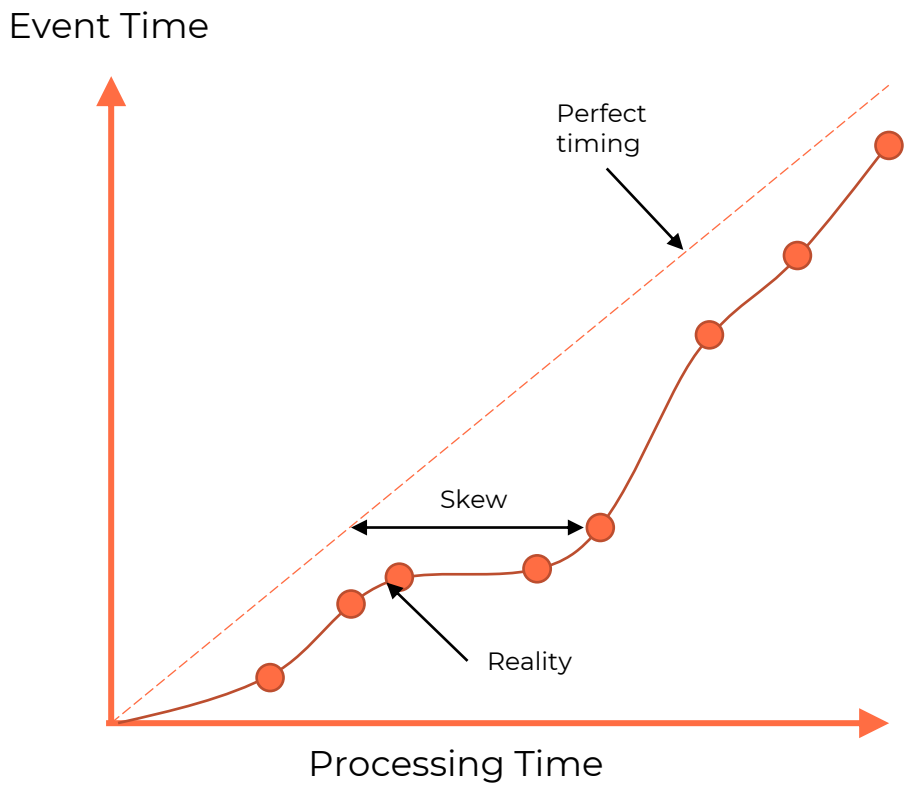
EVIDEN

Streaming processing

Time

What real example do you have with time skew ?

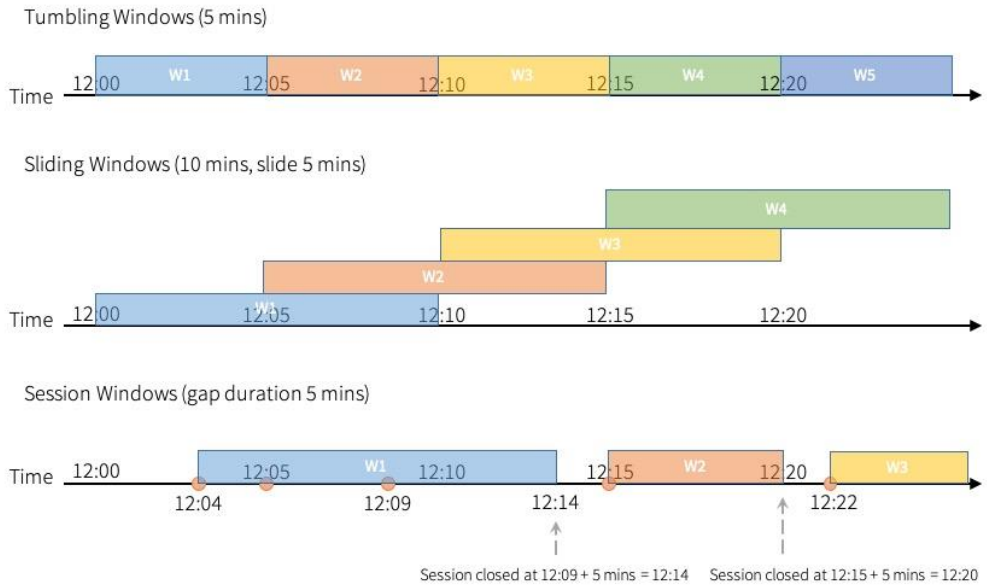
- **Event time**
  - Time of the event when it was generated by the operational system
- **Processing time**
  - Time when the event is received by the processing system
  - Note : in fact, this event could be processed some minutes latter due to windowing latency
- **Skew**
  - Difference between processing and event time
  - It’s not a useful metric for business but it has huge impact technically (see late events)

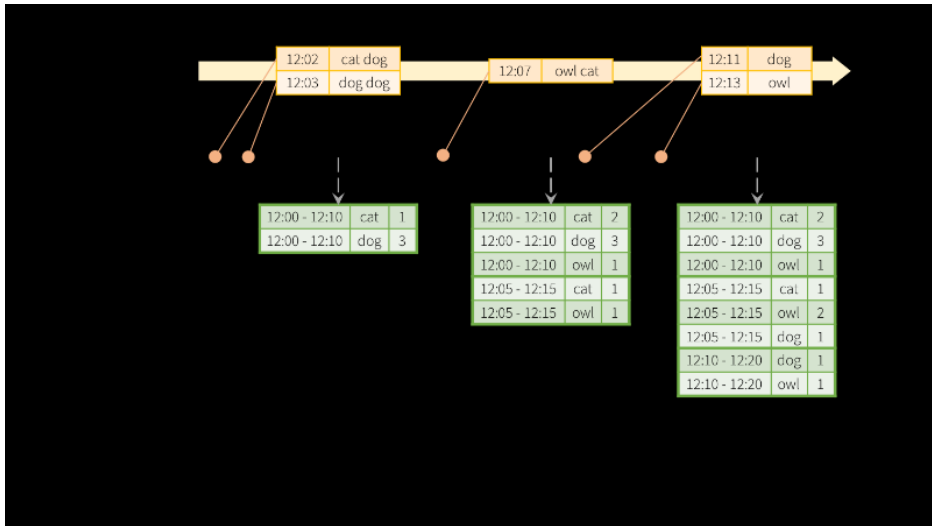


Streaming processing

Windowing

- **Without window** : forced to process every event unitary, no way to link them together, have counts, agregation, etc
- **Standard types of windows**
  - Fixed
  - Sliding
    - fixed size but overlapping, so be careful with “duplicated” information
  - **Session**
    - dynamic sized windows
    - if n min passes after the last received event without any new event, the window is closed



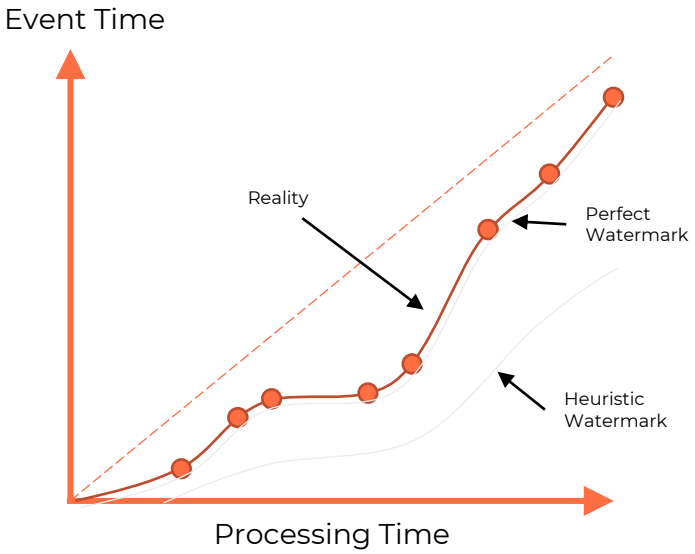


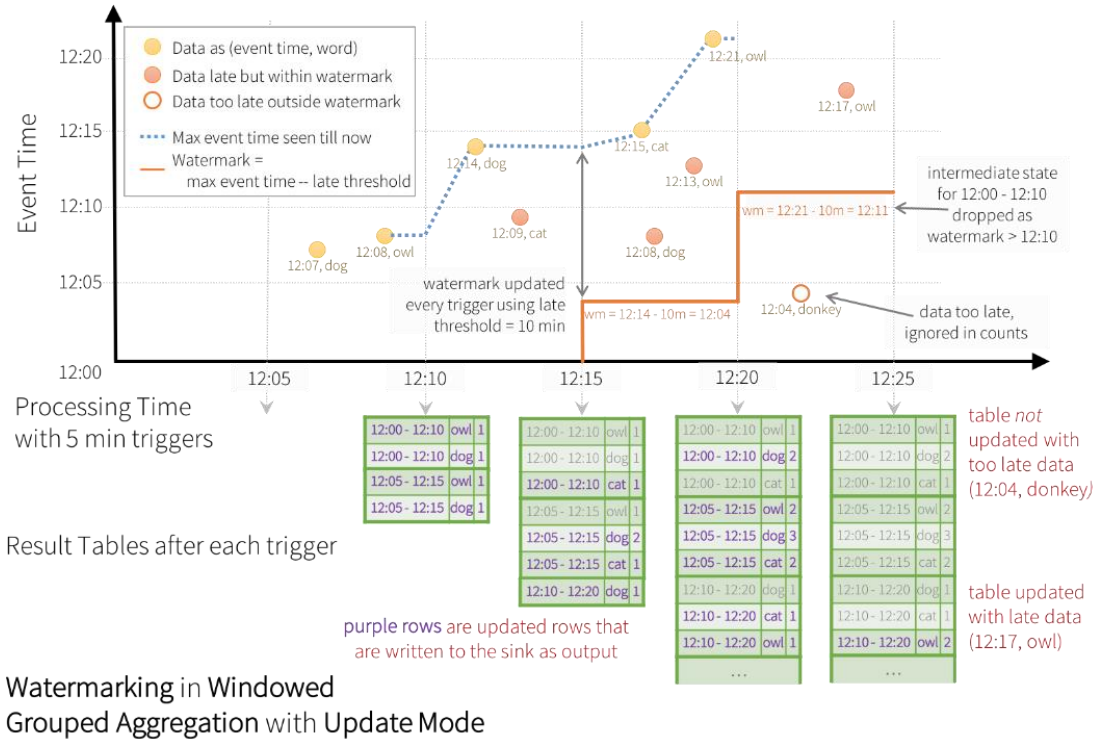
Warning, at 12:10, there’s not 3 cats, but 2 cat in the window 12->12:10 and 1 in the window 12:05->12:15

Streaming processing

Watermark

- Limit in event time after which we will not consider data
  - Perfect watermark : we have **perfect knowledge** of all the data (event time), no late data, all data are early or on time
  - Heuristic watermark : **estimation** of the better watermark based on metrics



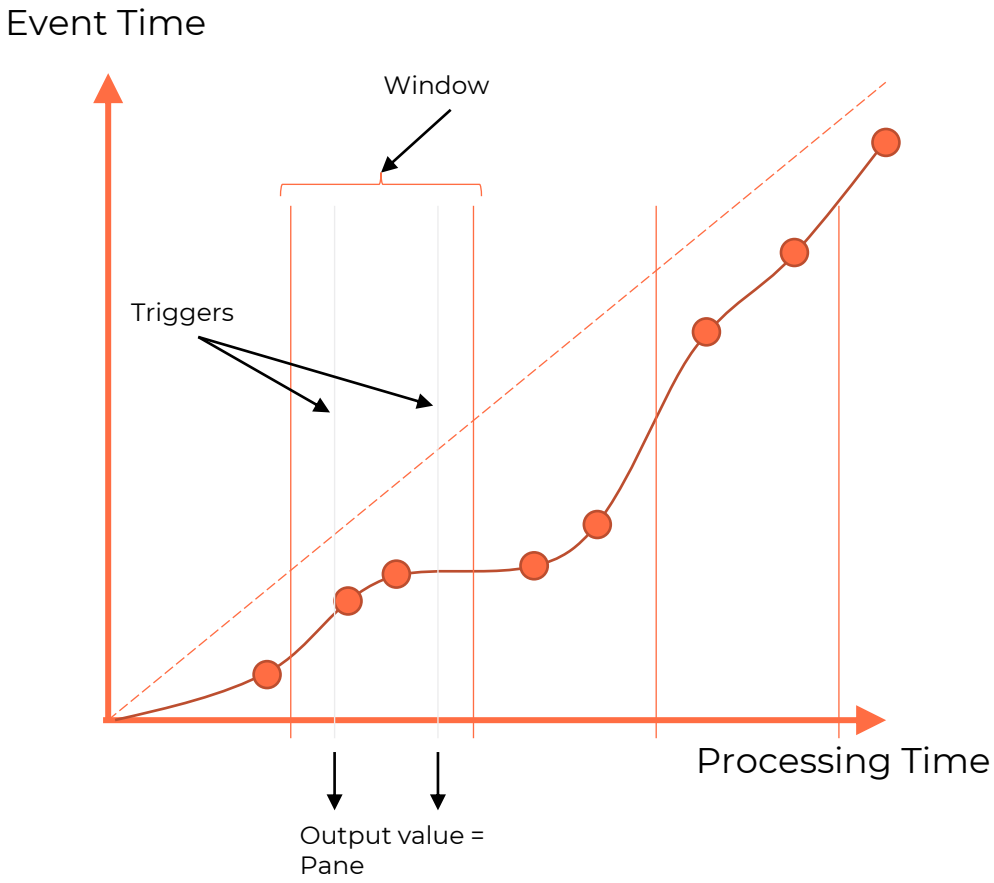


Watermak strategy is a balance between completeness and freshness : slow watermark will wait long time for late data, fast one will forget data

Streaming processing

Triggers

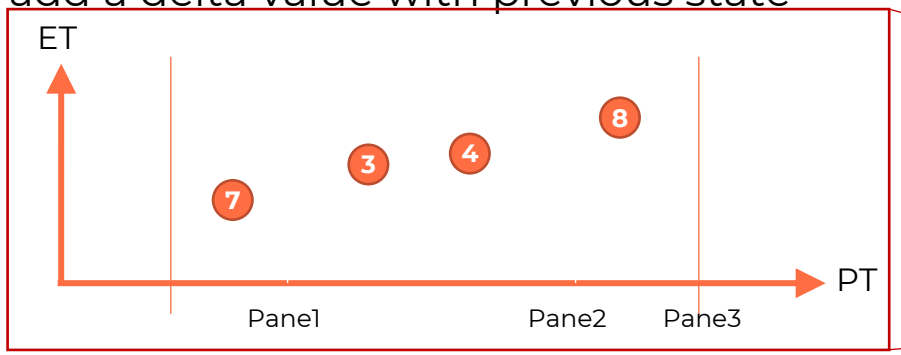
- Used to materialize result from window
- Can be based unpon simple actions
  - Watermark progress (=event time progress)
  - Processing time progress : periodic update within window
  - Element counts : after constant number of element have been observed in the window
  - Specific data in the event : after receiving special event
- Or more complex
  - Repetitions : useful with processing time progress
  - Logic (and, or, ...) : composition of triggers
  - Sequence : will fire child triggers sequentially

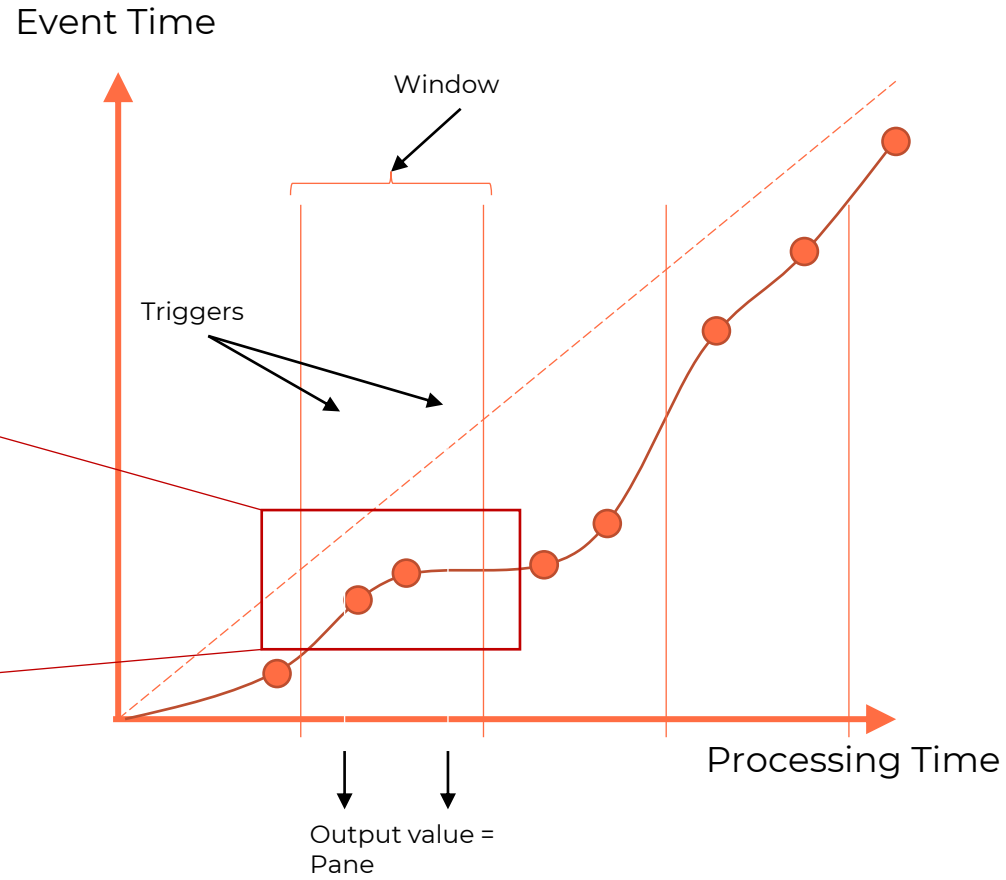


Streaming processing

Accumulation

- With multiple panes, how do the output is computed
  - Discard mode : only keep the current value
  - Accumulating : all previous panes states are retained and the current value is the addition of every previous states
  - Accumulating & retracting : like accumulative mode, plus add a delta value with previous state



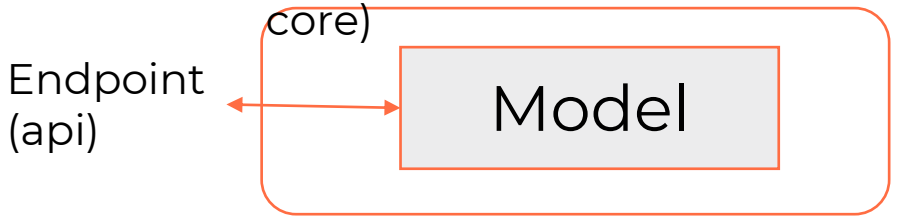


	Discarding	Accumulating	Accumulating & Retracting
Pane 1: [7]	7	7	7
Pane 2: [3, 4]	7	14	14, -7
Pane 3: [8]	8	22	22, -14
Last Value Observed	8	22	22
Total Sum	22	51	22

Inference graph

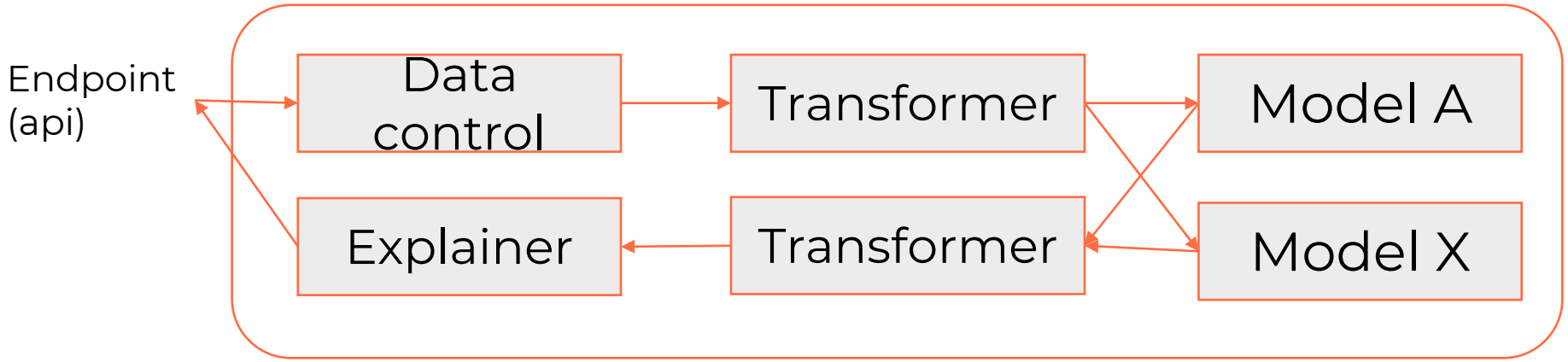
Event triggered ML specific services chain

Simple graph (model as core)



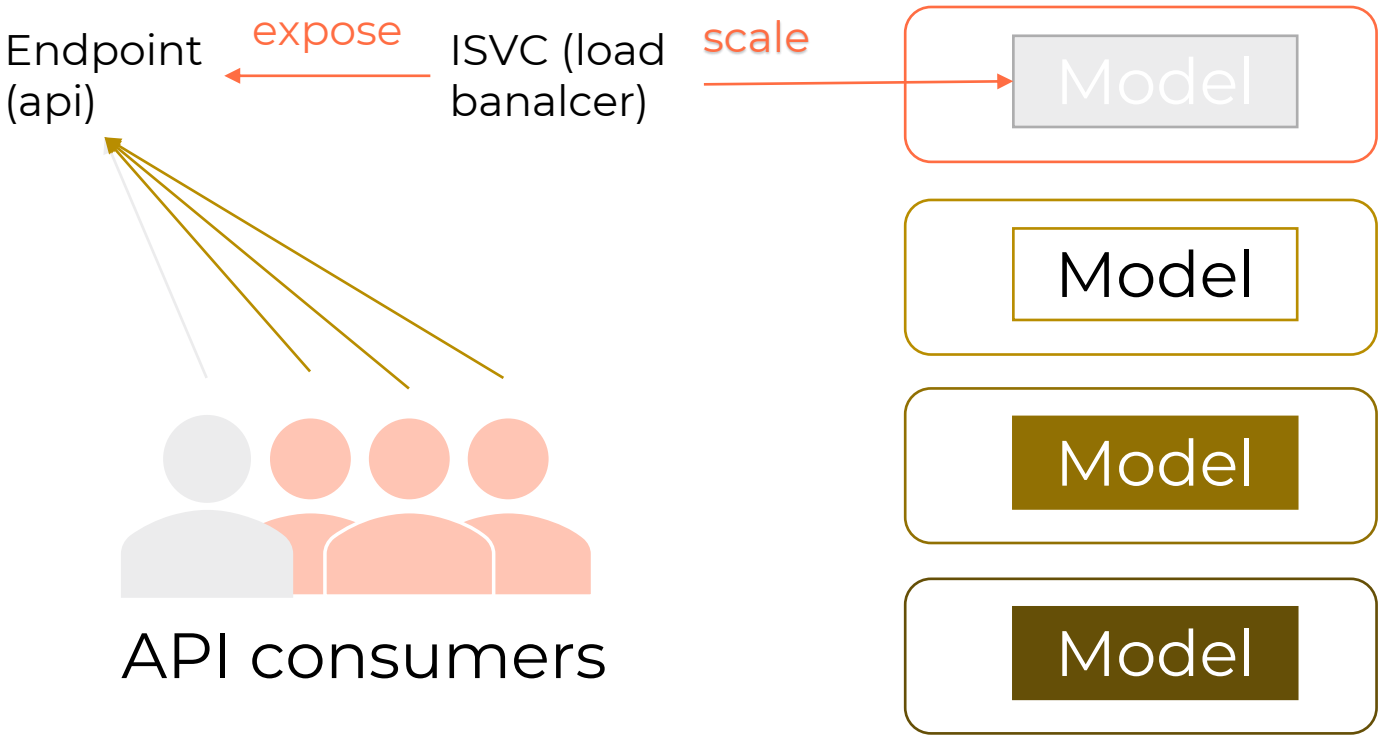
Exactly same feature engineering should be applied (inside or outside inference) than in training

Multi model Complex graph with Explainability and data control



Serving Auto scaling

As in software or data engineering, an inference service (ISVC) has to scale on demand



Downscale is as important as upscale for costs management

Explainability in inference graph

Online and Ad-hoc report of model behavior

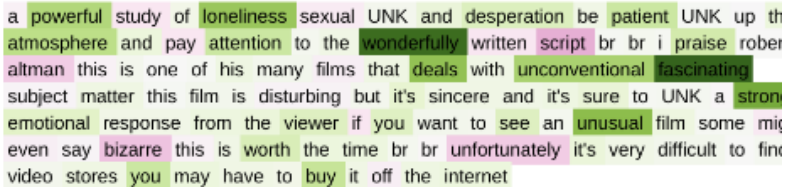
Nominal Case

Get sample insights and report it

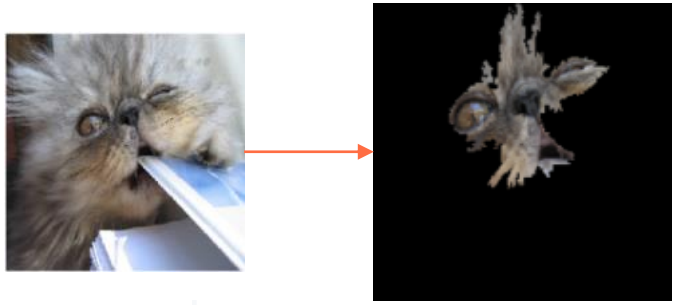
Anomaly case

Re-do prediction with handly control

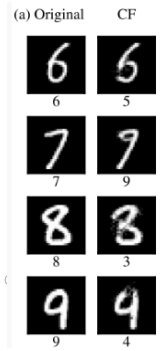
Integrated gradient for text classification



Anchor Explanation for images

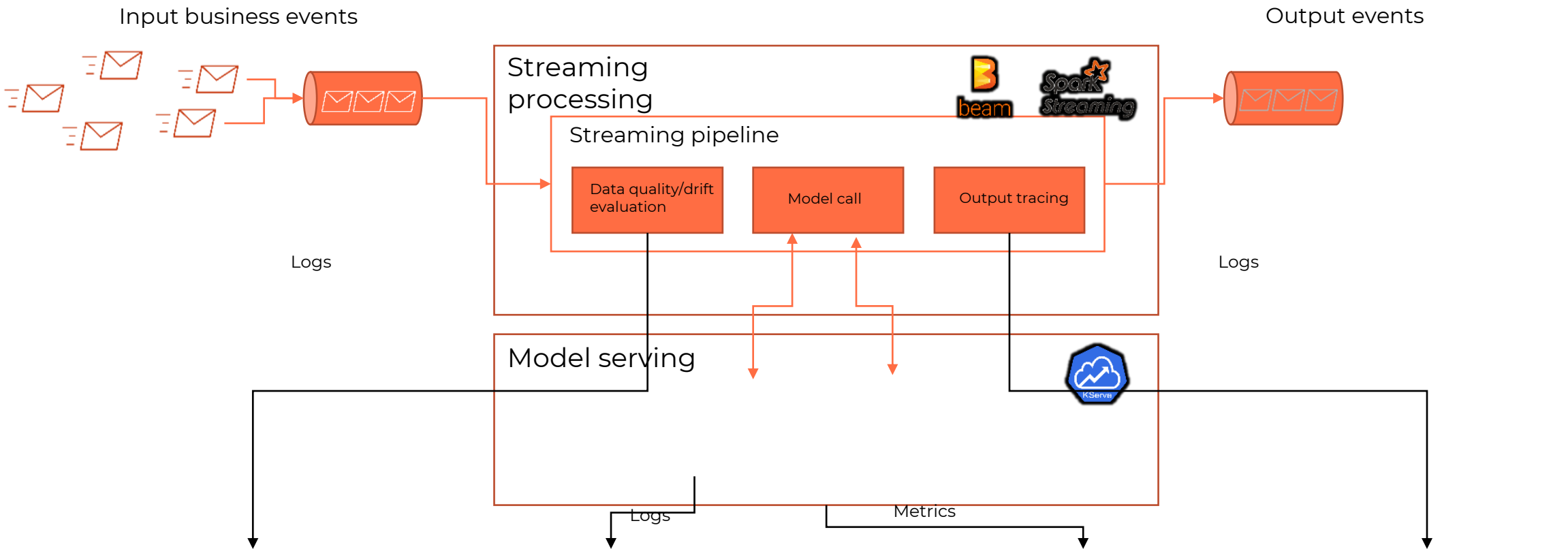


Counter factual examples (define what feature change affect the prediction)



Simple streaming monitoring

Add few steps in your pipelines



Quizz

What we've learned

Question				
Complex Event Processing is when we combine multiple events	Y	N		
Data coherence is the fact that every input event is well handled by the system	Y	N		
Event order is always granted by the streaming systems with "Exactly once" mode	Y	N		
Kappa architecture is a golden bullet to achieve state of the art architecture	Y	N		
In an Event Driven Architecture, Ops Forge (registry, CICD, ...) is still a necessary component	Y	N		
In an Event Driven Architecture model serving and data processing is answered by the same technology	Y	N		
Processing time and event time are the same in a perfect timing	Y	N		
Processing time can be younger than event time (p_time < e_time)	Y	N		
Triggers in streaming processing are linked with data output of a window	Y	N		
Transformers are a way to do data processing in inference graphs	Y	N		
Is model serving scaling rely on usage metrics and availability of infrasctructures ?	Y	N		
Event monitoring is only based on log management	Y	N		

Quizz

What we've learned

Question				
Complex Event Processing is when we combine multiple events	Y	N		
Data coherence is the fact that every input event is well handled by the system	Y	N		
Event order is always granted by the streaming systems with "Exactly once" mode	Y	N		
Kappa architecture is a golden bullet to achieve state of the art architecture	Y	N		
In an Event Driven Architecture, Ops Forge (registry, CICD, ...) is still a necessary component	Y	N		
In an Event Driven Architecture model serving and data processing is answered by the same technology	Y	N		
Processing time and event time are the same in a perfect timing	Y	N		
Processing time can be younger than event time (p_time < e_time)	Y	N		
Triggers in streaming processing are linked with data output of a window	Y	N		
Transformers are a way to do data processing in inference graphs	Y	N		
Is model serving scaling rely on usage metrics and availability of infrasctructures ?	Y	N		
Event monitoring is only based on log management	Y	N		

"Exactly once" is a delivery mode, not a data coherence tool
Kappa Architecture is not a golden bullet, theire's some flaws (eg: every data is not always good to be pushed in events)
Usually model serving and data processing can be awsered by 2 different components/technologies
When we deal with future business events, event time is older than processing time
Heuristic watermarks are doing their best to wait for late data, but some time late data are too late...
XAI can be used in lab/training phase and in inference phase
Monitoring is based on metrics comming from logs and infrastructure

In Practice

Lab Content

- Exo1 Upgrade taxi tips Pipeline
  - Create a preprocessing component to clean data before training
  - Create a model export component to persist your model in MinIO
  - Adapt the pipeline to use those 2 components and run it to export a trained model to minio
- Exo2 Kserve inference service
  - From the persisted model, create an inference service with kserve client
  - Built the correct call to query this model from the notebook

- Go further :
  - Deploy the Inference service from a pipeline component

EVIDEN