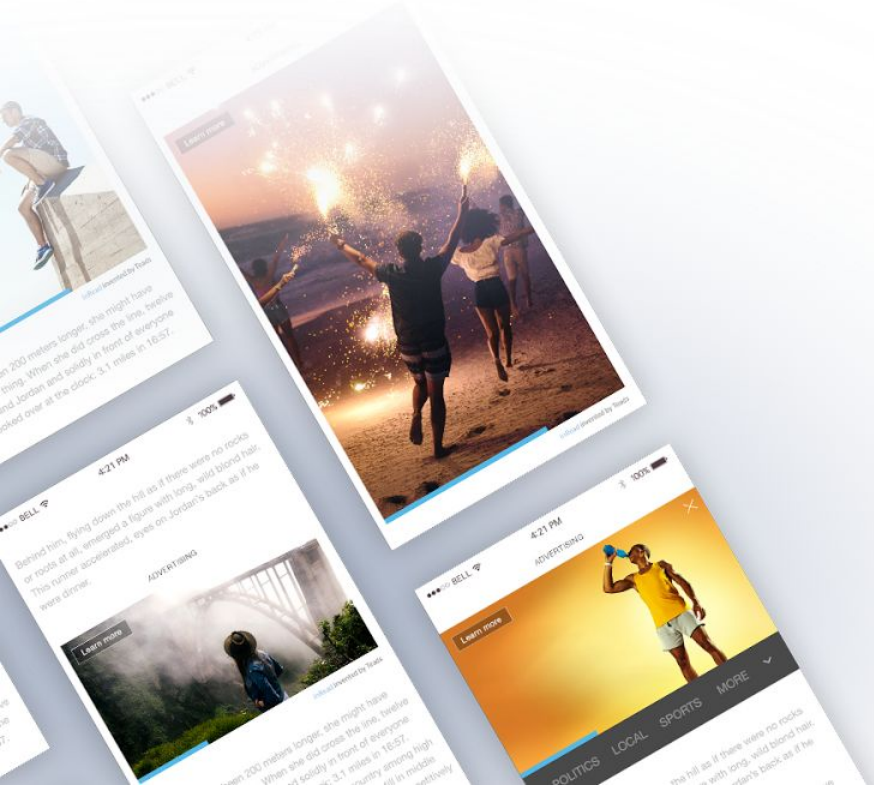


Teads.tv

Reinventing Video Advertising

Teads Analytics Stack

Keeping up with exponential growth

27 offices
in 21 countries

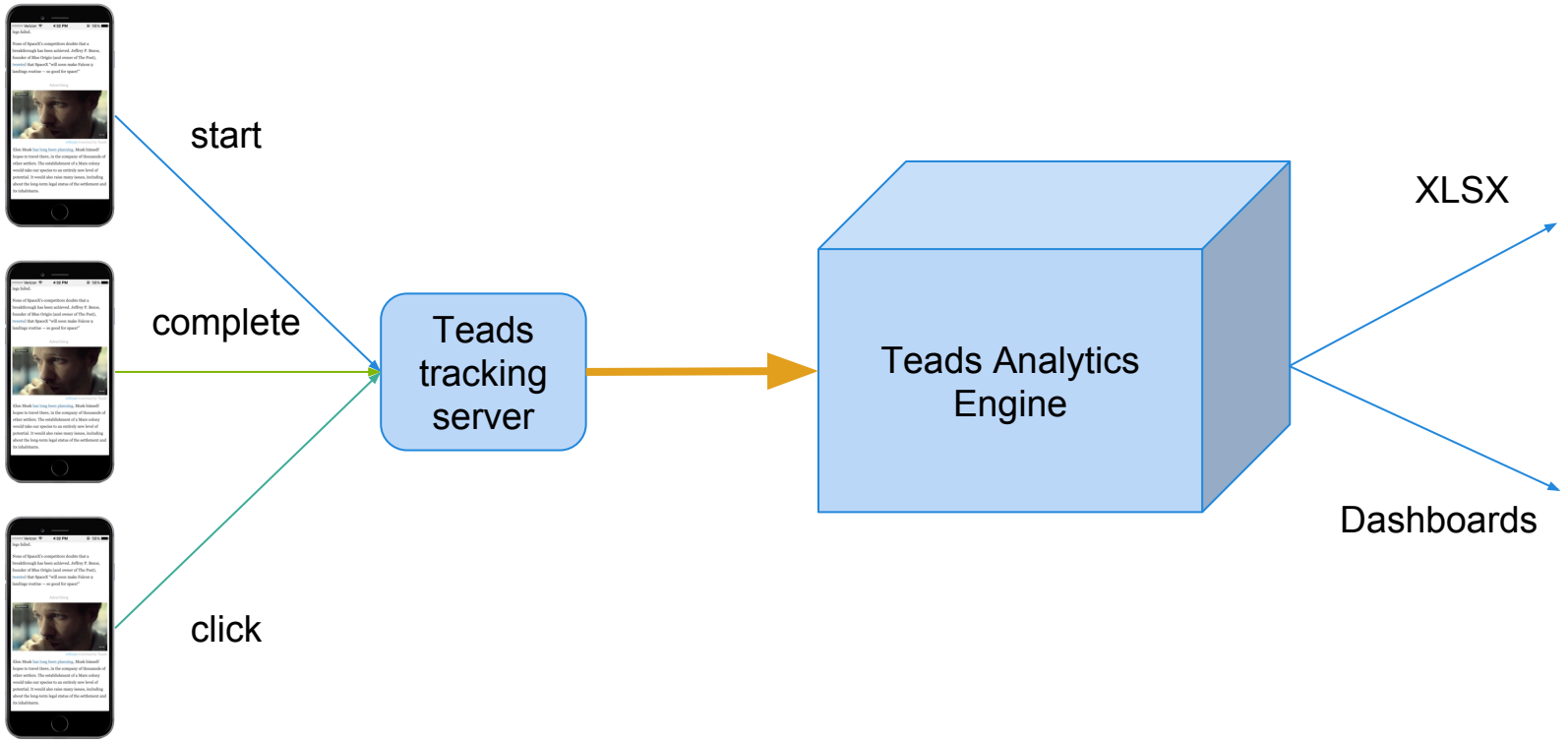
$200M
2016 Revenue

500+
Globally employees

90+
R&D employees

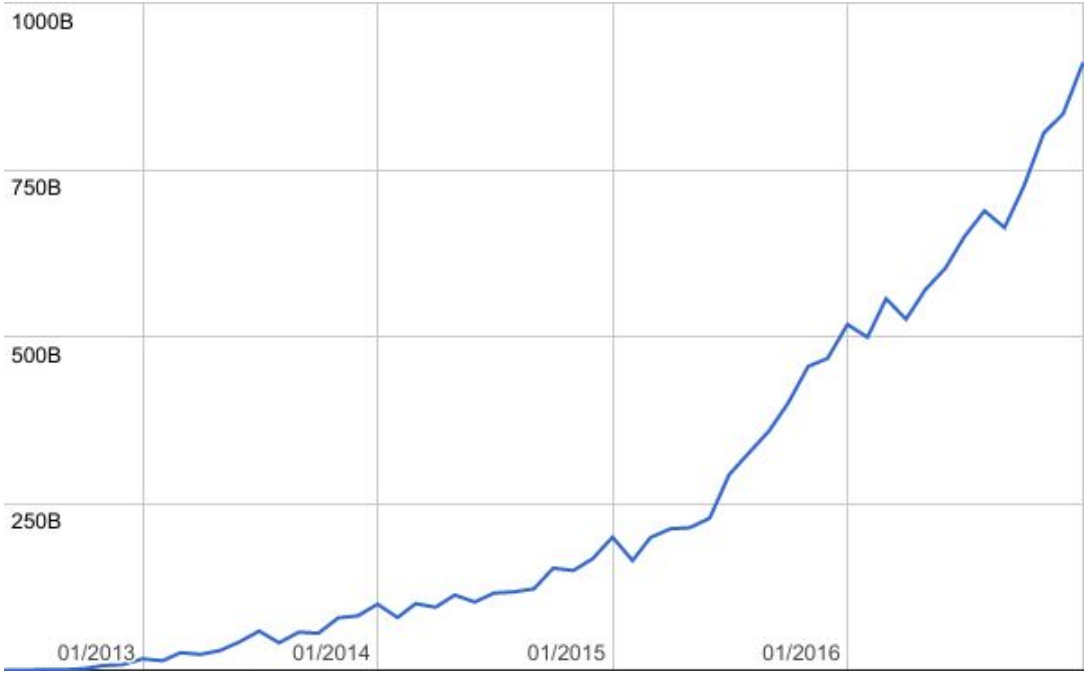
Teads.tv

From a product to data engineering



Teads Growth

Tracking events
Per month



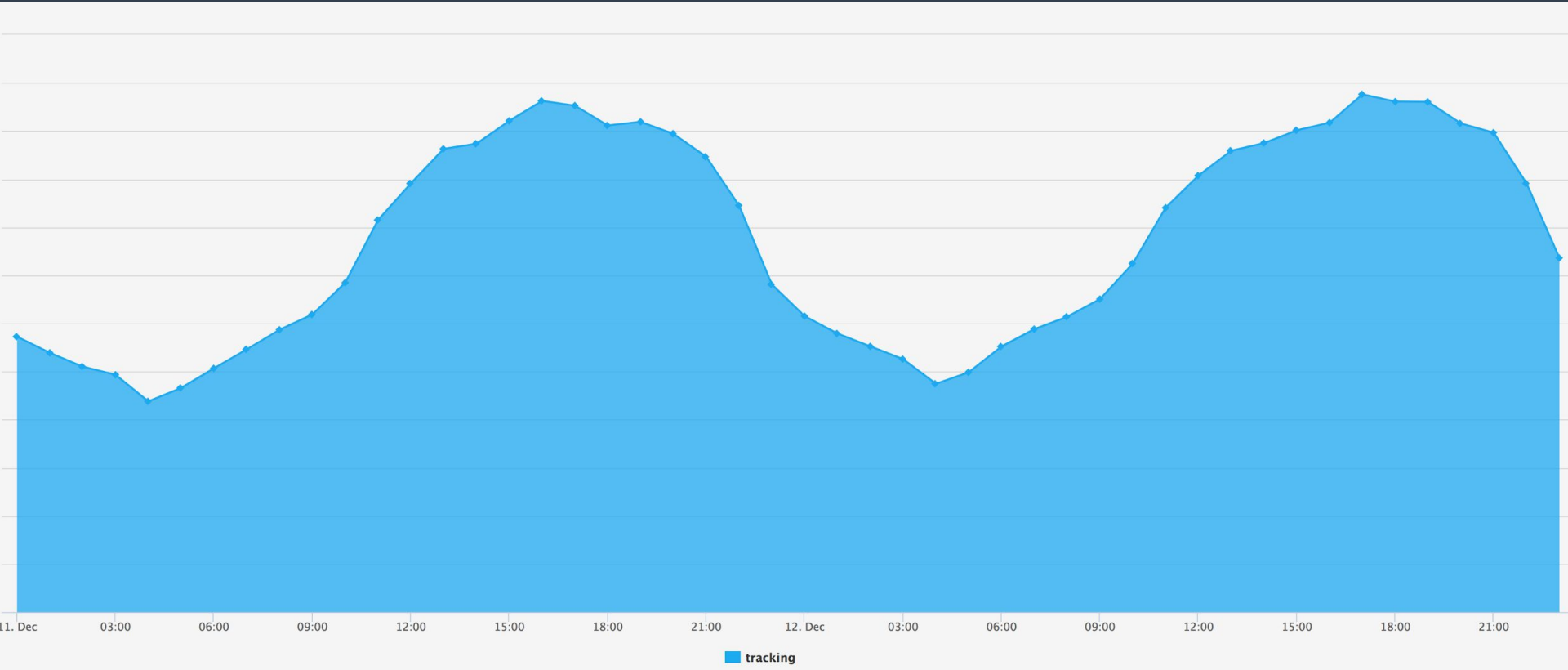
Analytics scale at Teads

- December 2013
  - 3000 tracking events per second on average
  - 2 GB of logs per day

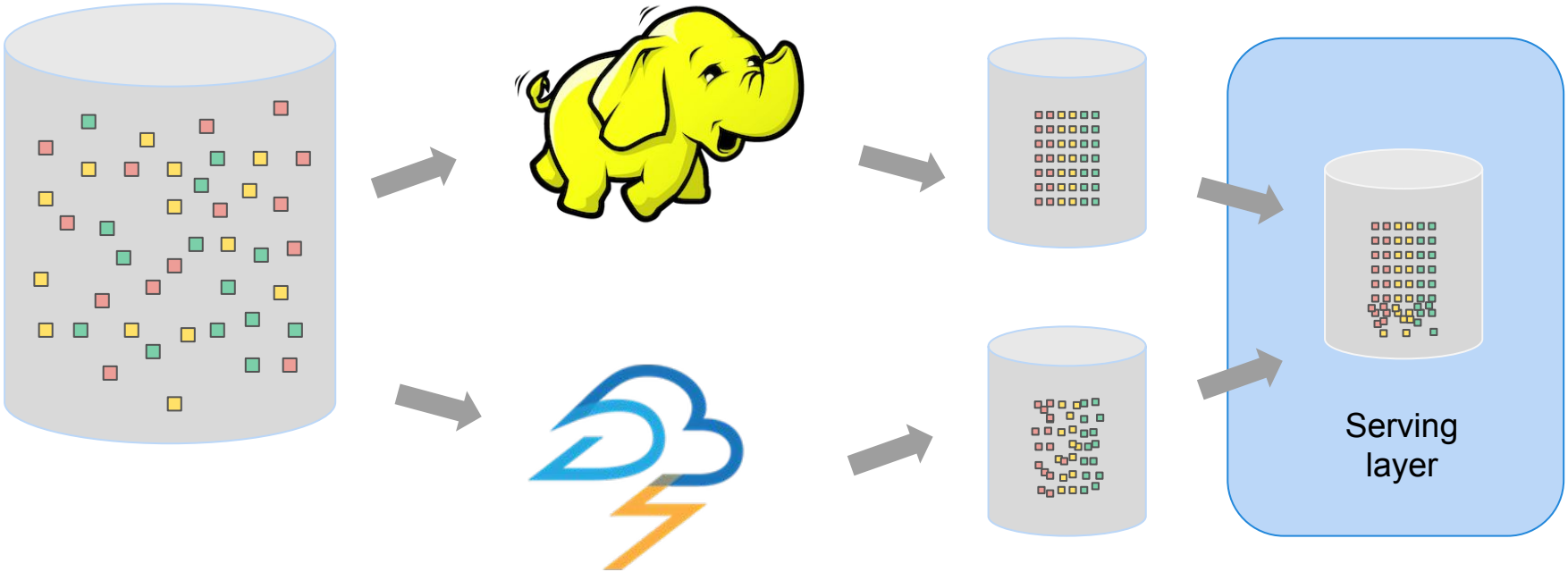
- December 2017
  - 300 000 tracking events per second on average (80 000 after sampling)
  - 10 TB of raw logs per day (2 TB after sampling)

Teads.tv

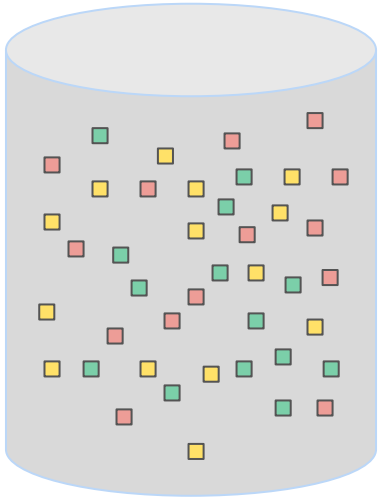
Event trend


11. Dec
03:00
06:00
09:00
12:00
15:00
18:00
21:00
12. Dec
03:00
06:00
09:00
12:00
15:00
18:00
21:00
tracking

Teads Lambda architecture (2014 - 2016)

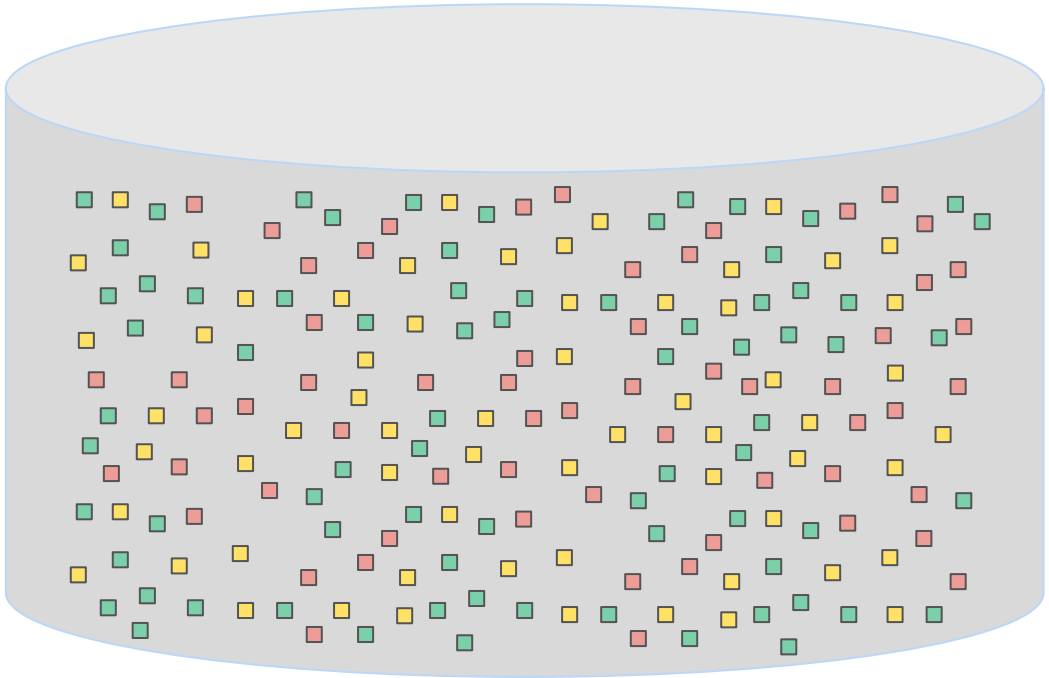


Data ...



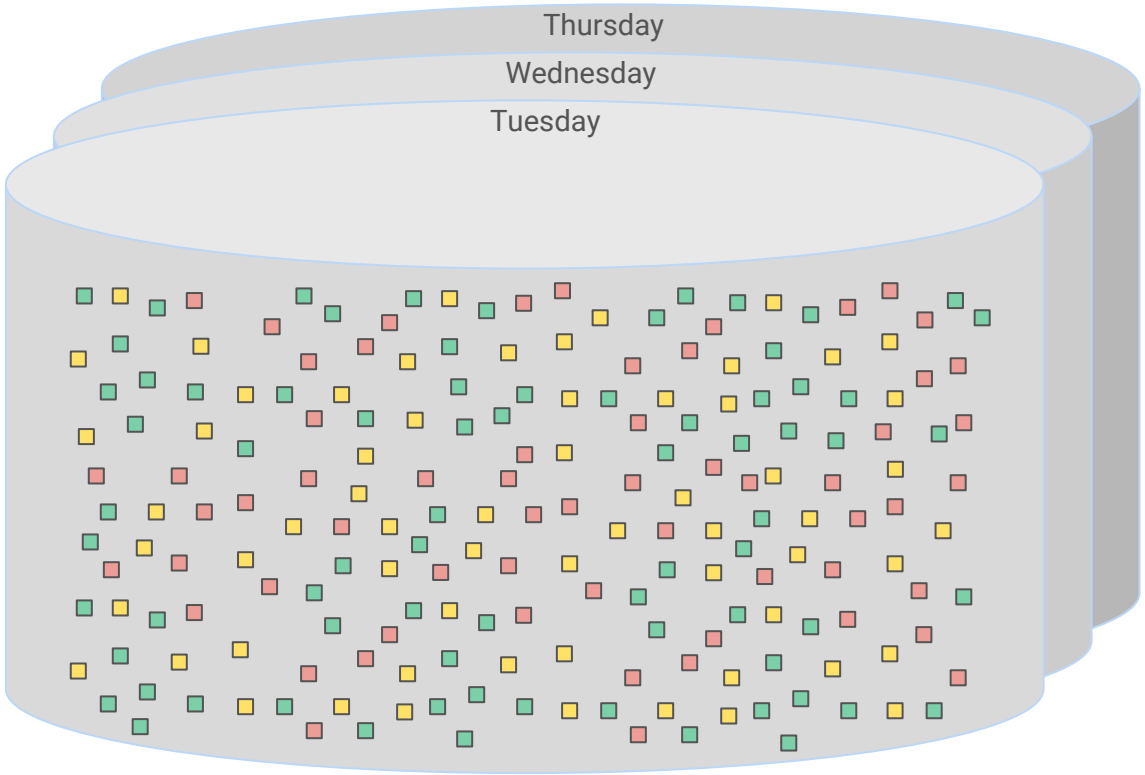
Teads.tv

… can be big …

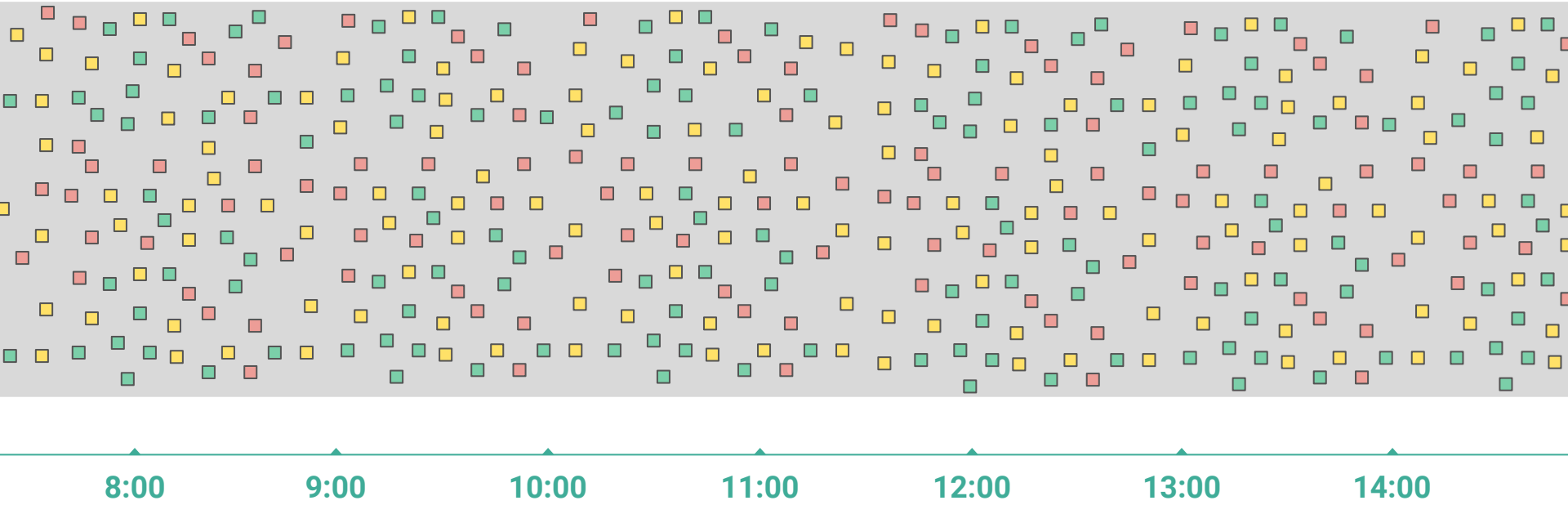


Teads.tv

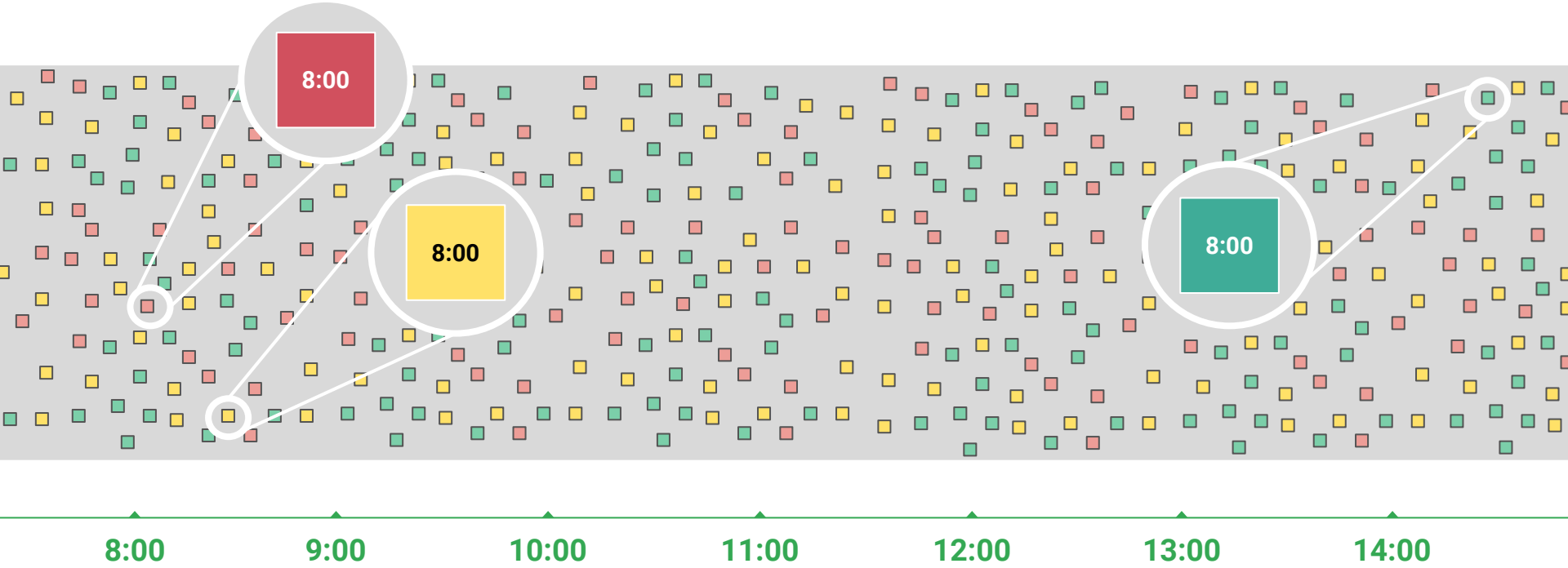
… really, really big …



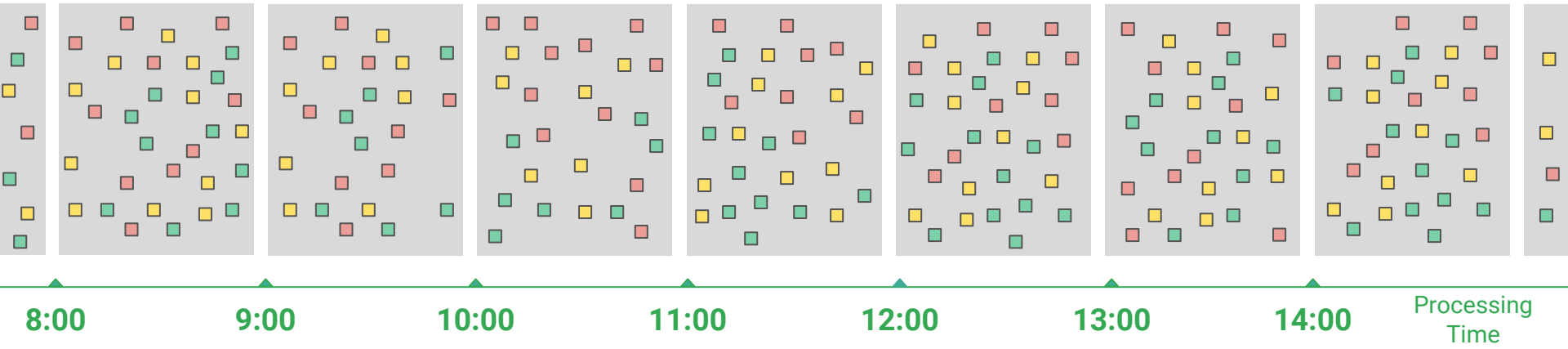
… maybe infinitely big …



… with unknown delays

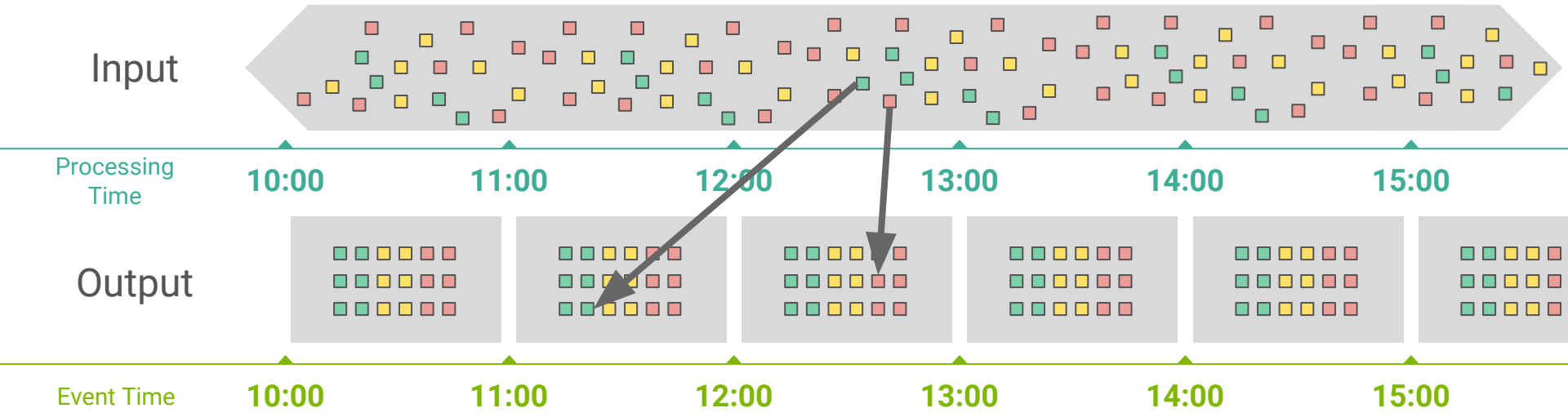


Aggregating via processing-time windows

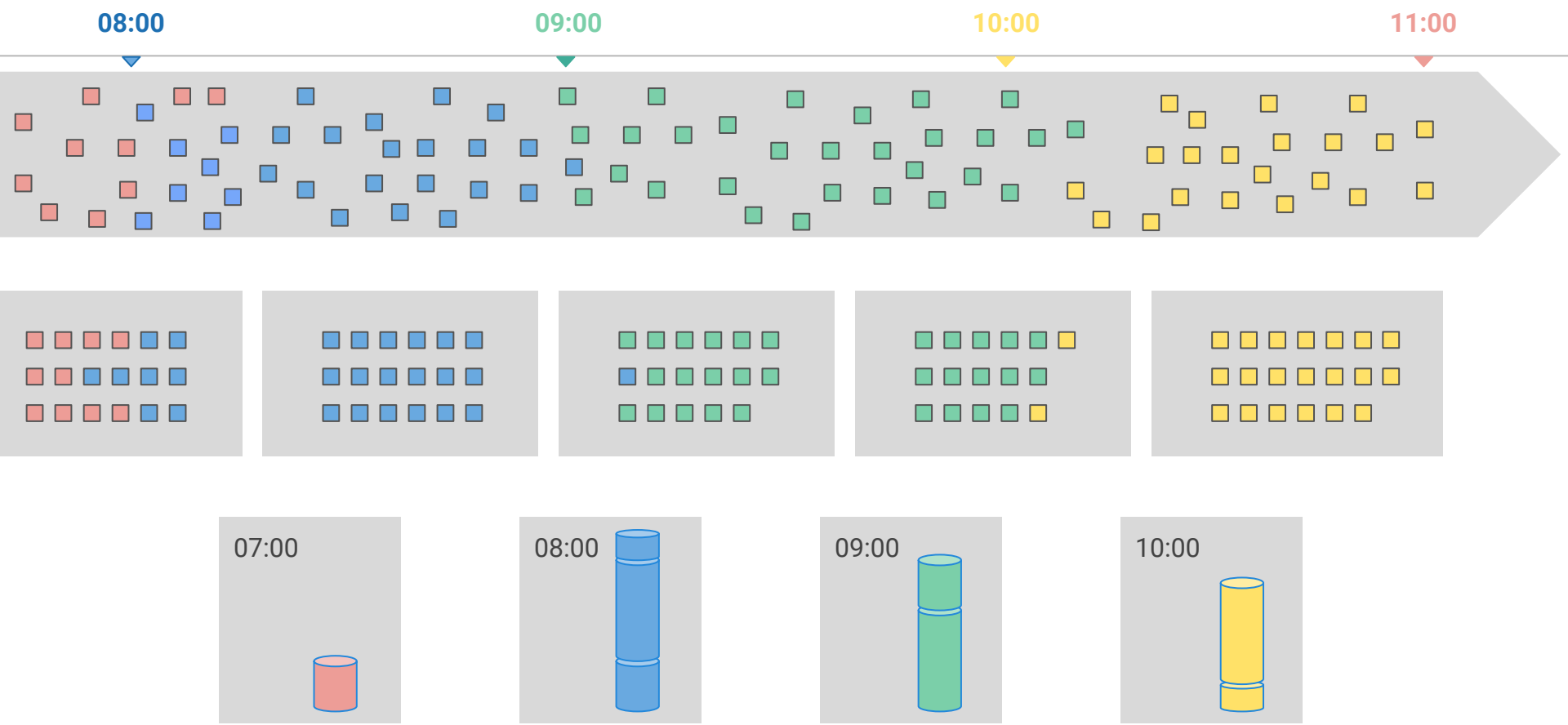


Teads.tv

Aggregating via event-time windows



Teads.tv


08:00
09:00
10:00
11:00
07:00
08:00
09:00
10:00

Moving to a new paradigm

- Scalable
  - What if the traffic doubles?
- Cost efficient
  - Can we keep costs linear?
- “Easy to manage”
  - Do we need to manage versions ourselves?
  - Is the service able to recover on its own? Is it properly monitored?

Teads.tv

Moving to a new paradigm

- Reproducible
  - What if there is a bug/crash in our software and we need to reprocess?
- Acceptable latency
  - How long does an event takes to go through the whole data pipeline?
- Avoid code duplication
  - How can we can the code factored?

Teads.tv

Teads Analytics choice

Dataflow

+

BigQuery

Kappa Architecture

Teads.tv

Kappa architecture practically

- Short batches
  - ~ 30 minutes
- Each batch is between Kafka offsets
  - It’s reproducible
- Production friendly
  - “Based” on Apache BEAM (Open source)
  - Processing in Scala (based on SCIO from Spotify)
  - Easily scalable (it works with on demand workers per job)

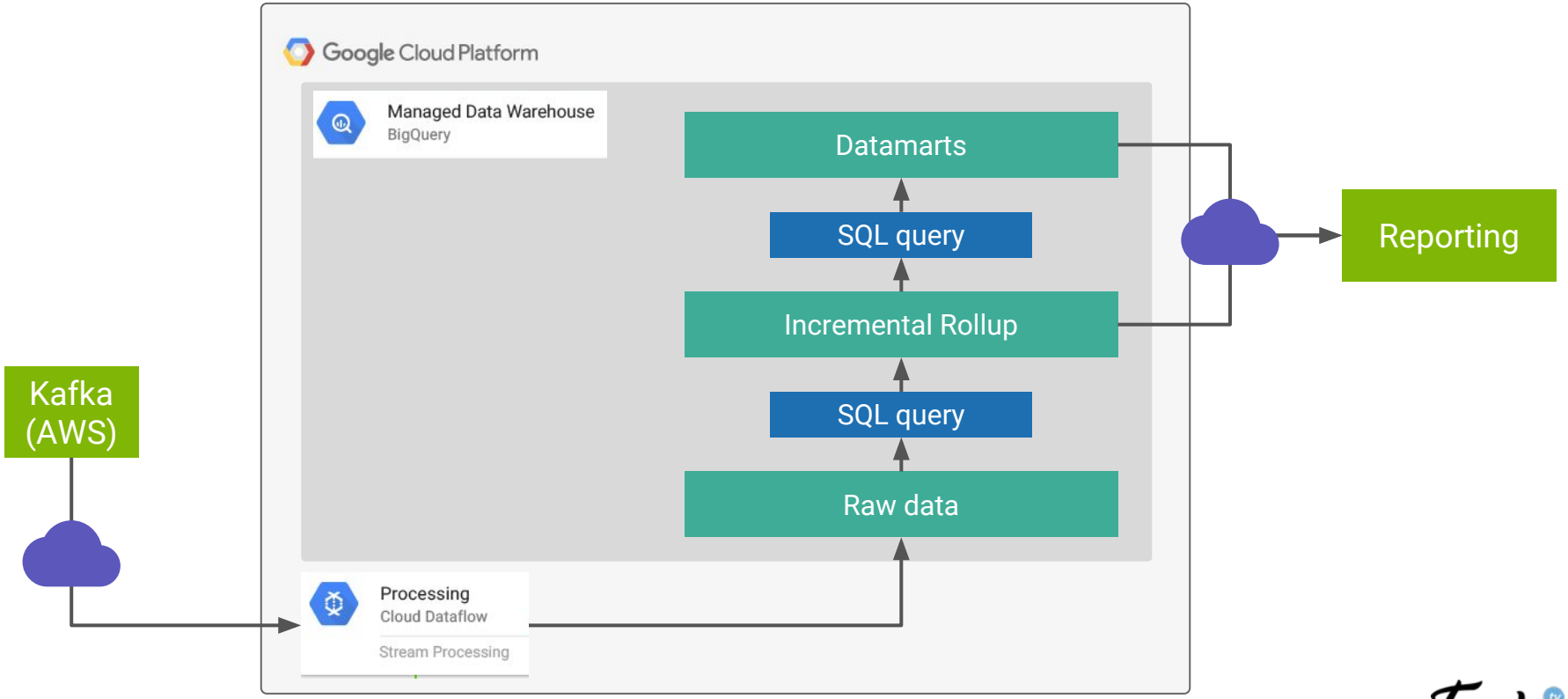
Teads.tv

Kappa architecture practically

- Fully SQL compliant
  - Easy to write powerful **ELT** jobs
  - Product managers / support engineers can explore/troubleshoot data
- Operations are atomic
  - You never end up with partial data, it’s either there or not
- Convenient pricing plans
  - Flat pricing, you pay 40k$/month for 2000 slots
  - Storage and processing power are not correlated

Teads.tv

Architecture schema



Rollup

Time	Ad ID	Website ID	Event
1465891280	1550	12820	impression
1465891281	1550	12820	start
1465891282	1550	12820	click
1465891284	1550	12820	impression

Hour	Count	Ad ID	Website ID	Event
10	2	1550	12820	impression
10	1	1550	12820	start
10	1	1550	12820	click

Teads.tv

Rollup

Hour	Count	Ad ID	Website ID	Event
10	2	1550	12820	impression
10	1	1550	12820	start
10	1	1550	12820	click

Hour	Ad ID	Website ID	impression	start	click
10	1550	12820	2	1	1

Teads.tv

Columnar storage

Hour	Ad ID	Website ID	impression	start	click
10	1550	12820	2	1	1
10	1550	12823	4	4	2
10	1550	12838	2943	2943	1235

Teads.tv

Hourly tables

Table 2017121310

Ad ID	Website ID	impression	start	click
1550	12820	2	1	1
1550	12823	4	4	2
1550	12838	2943	2943	1235

Teads.tv

Advantages

- Much smaller rollups
  - Raw data vs rollups ratio is 70x
  - Way faster to request
- Easier to request
  - SELECT SUM(impression) as impression
    FROM rollup_hourly.2017121300
    WHERE ad_id = 123

Teads.tv

Inconveniences

- What about adding a new metric?
  - BigQuery doesn’t let users request multiple tables that have different schemas (even if it’s not using the fields that differ)
- How to request multiple hours?

Teads.tv

Our workarounds (1/2)

- Storing “custom” metrics in a nested array

Custom metrics		Ad ID	Website ID	impression	start	click
name	count					
		1550	12820	2	1	1
		1550	12823	4	4	2
fb	100	1550	12838	2943	2943	1235
twitter	50					

Teads.tv

Our workarounds (2/2)

- We use a select on UNION ALL to avoid schema issues
  - SELECT ad_id, SUM(impression) AS impression
    FROM (
    SELECT ad_id, impression FROM
    `rollup_hourly.2017121300`
    UNION ALL
    SELECT ad_id, impression
    FROM
    `rollup_hourly.2017121301`)
    GROUP BY ad_id

Teads.tv

Things we'd love to have

- Hourly partitioned tables
  - Right now, only daily partitioned table works
- BigQuery quotas
  - Being able to restrict the amount of slot used per person / service / team
- Federated source queries
  - Being able to join with Google Datastore, Google Cloud SQL, external SQL compliant databases, ...

Teads.tv

Thank you for your attention