End-to-end Exactly-once Aggregation over Ad Streams

Amiraj Dhawan

Amit Ramesh

Data Council

Yelp’s Mission

Connecting people with great
local businesses.

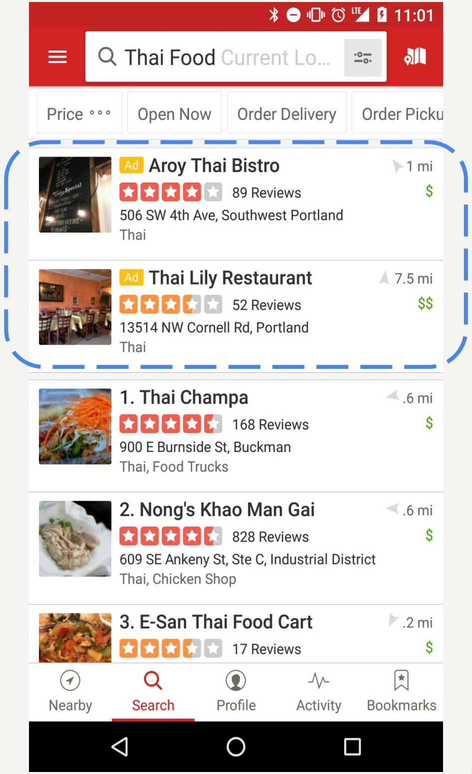
Outline

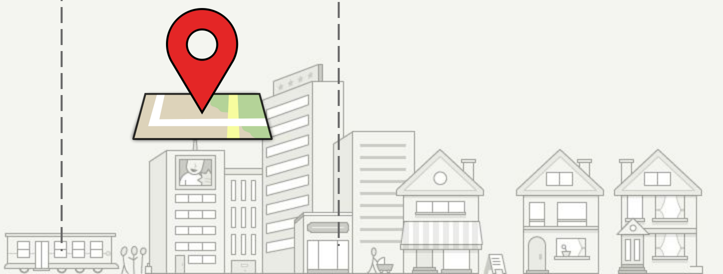
- Background & context
- Business requirements
- Design iterations
- Exactly-once aggregation
- What's next?

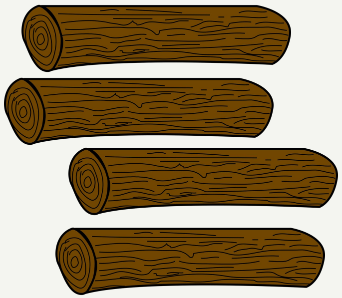
Local Ads

- Work done within the Local Ads group
- Manage a few 100K ad campaigns daily
- Mom and pop stores to national chains
- Pipelines receive a few thousand msgs/sec
- Pipelines in production for more than a year

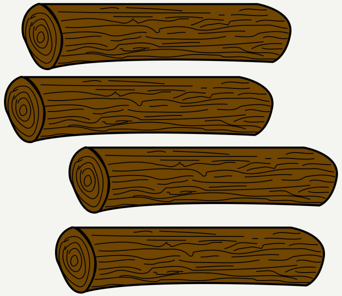
Local Ads – Consumer facing

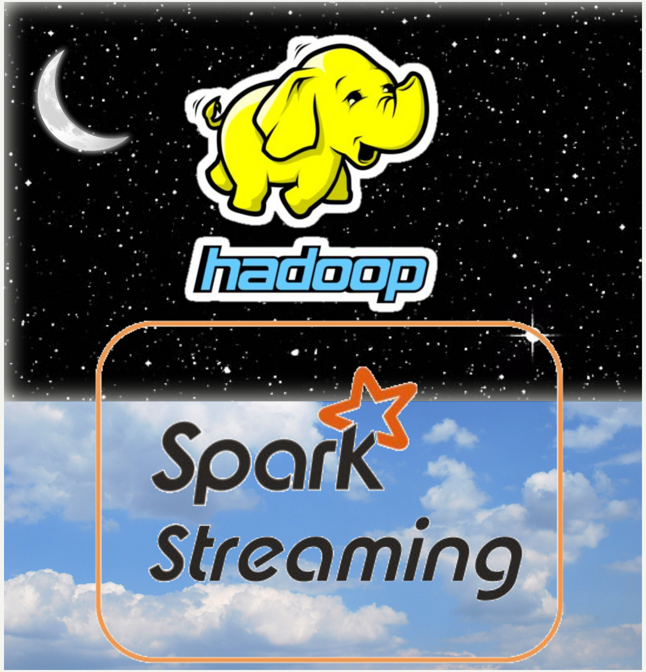


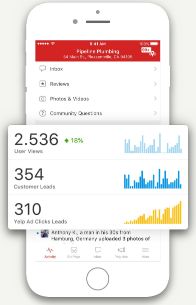




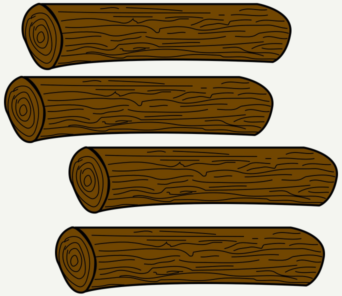
Local Ads – Advertiser facing







Local Ads – Ad Campaign Management



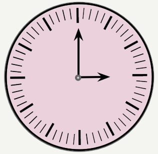
Distilled Business Requirements

- Aggregate events over a day period
- Slice aggregates along defined dimensions
- Provide partial aggregates as day progresses
- Make aggregates as accurate as possible

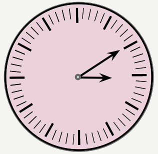
Day	Dimension 1	Dimension 2	Dimension N	Aggregate 1	Aggregate 2	Aggregate M

An Illustrative Example

- Count ad clicks over a day period
- Provide click counts by ad campaign
- Provide partial click counts as day progresses

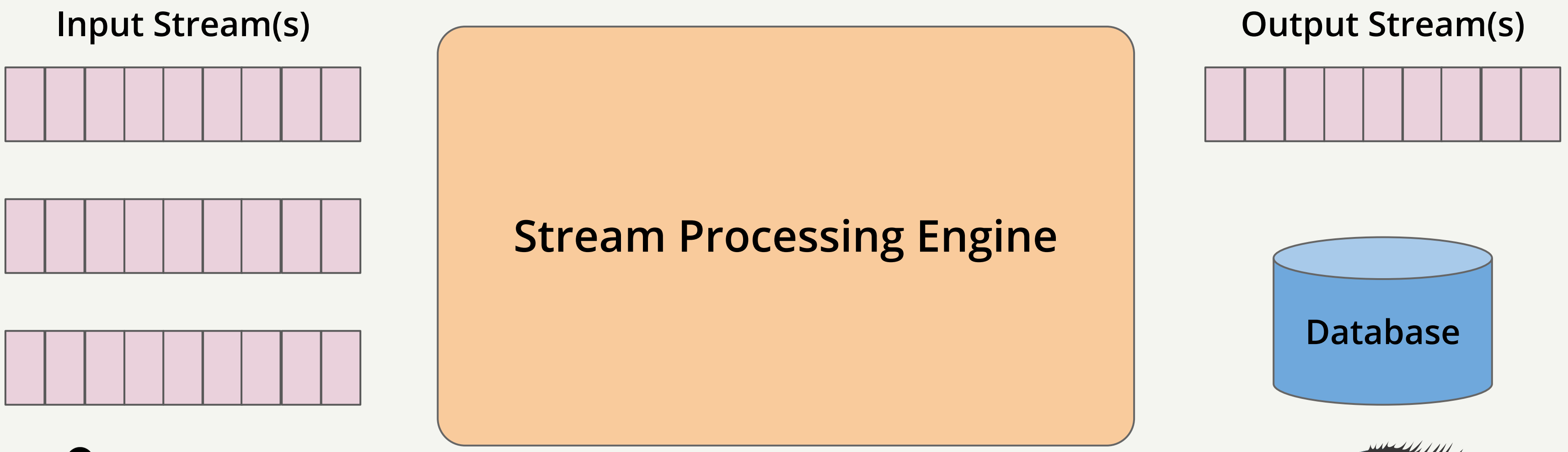


Day	Campaign ID	Number of clicks
4/17/2019	23265	35



Day	Campaign ID	Number of clicks
4/17/2019	23265	42

Stream Processing 101



APACHE kafka

amazon KINESIS

Spark Streaming

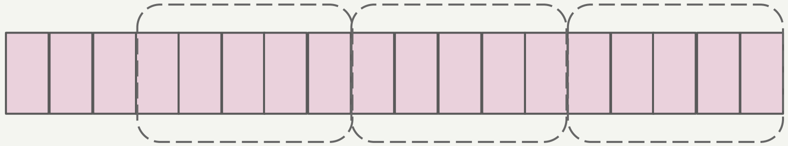
Flink

cassandra

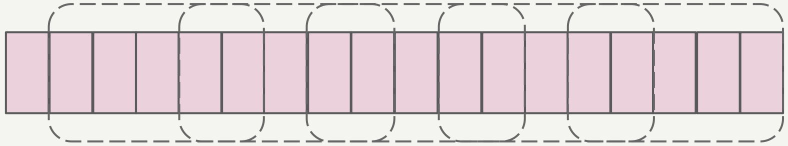
redis

Stream Processing 101

Windowed operations

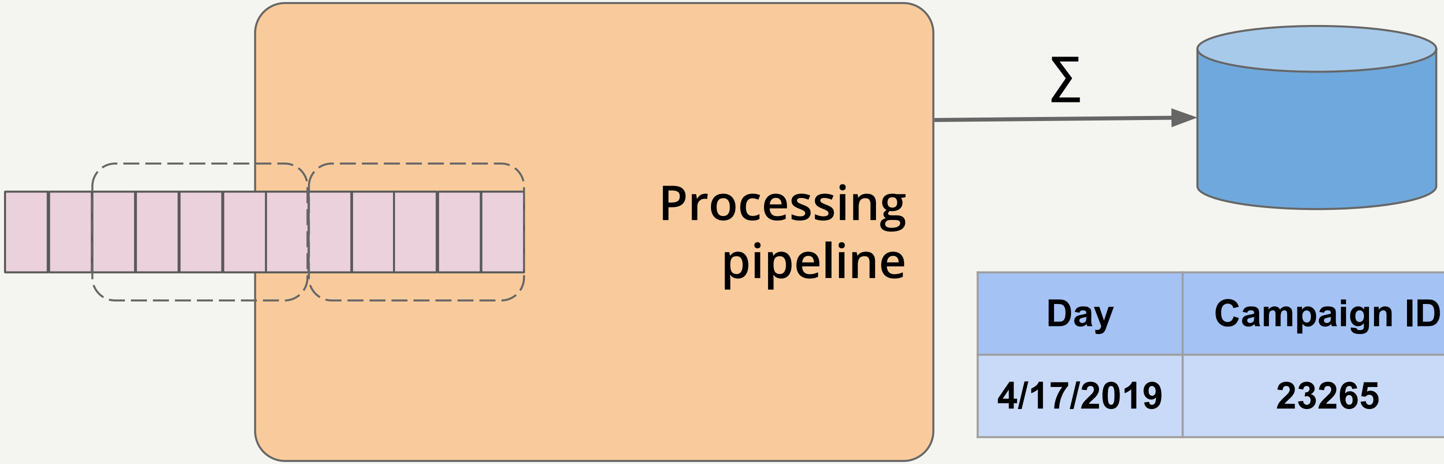


Tumbling window



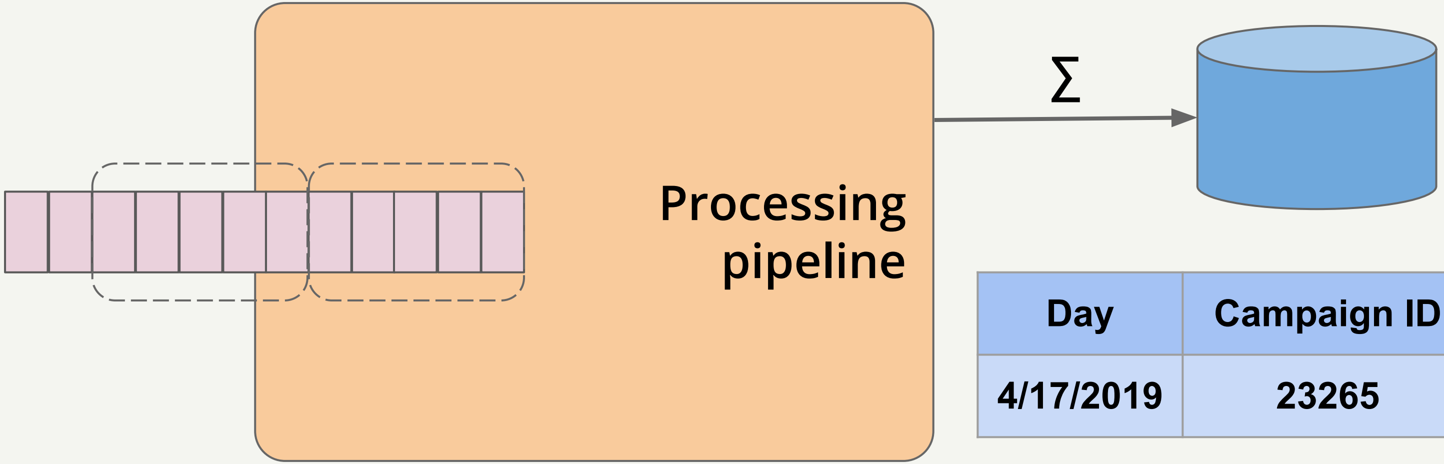
Sliding window

Why not...



Day	Campaign ID	Number of clicks
4/17/2019	23265	35

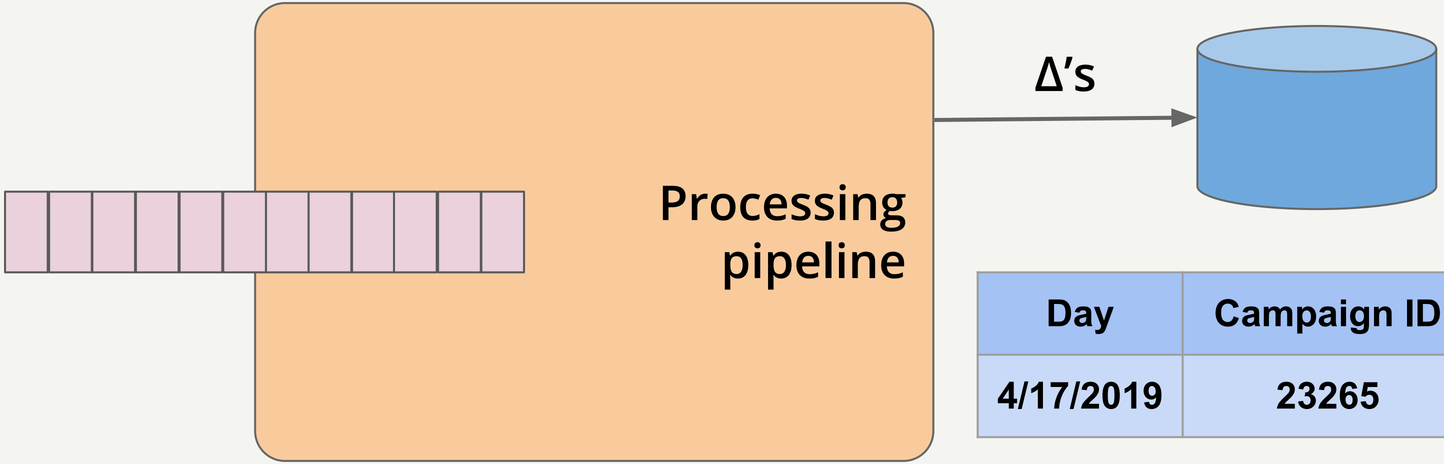
Why not...



Day	Campaign ID	Number of clicks
4/17/2019	23265	35

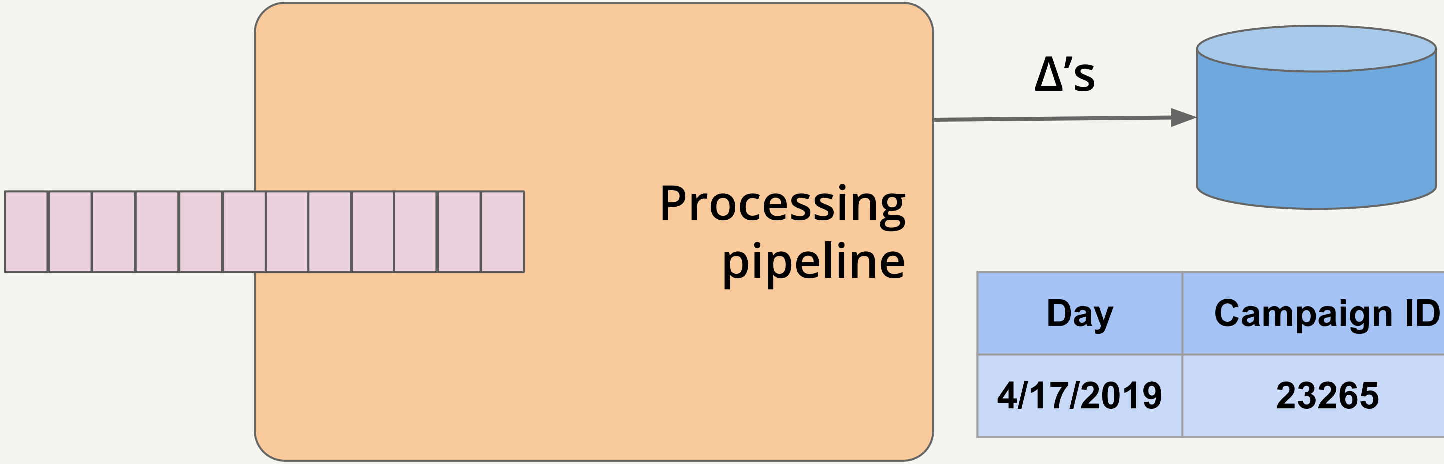
- Need partial click counts as day progresses!
- Stateful operation

How about...



Day	Campaign ID	Number of clicks
4/17/2019	23265	35

How about...



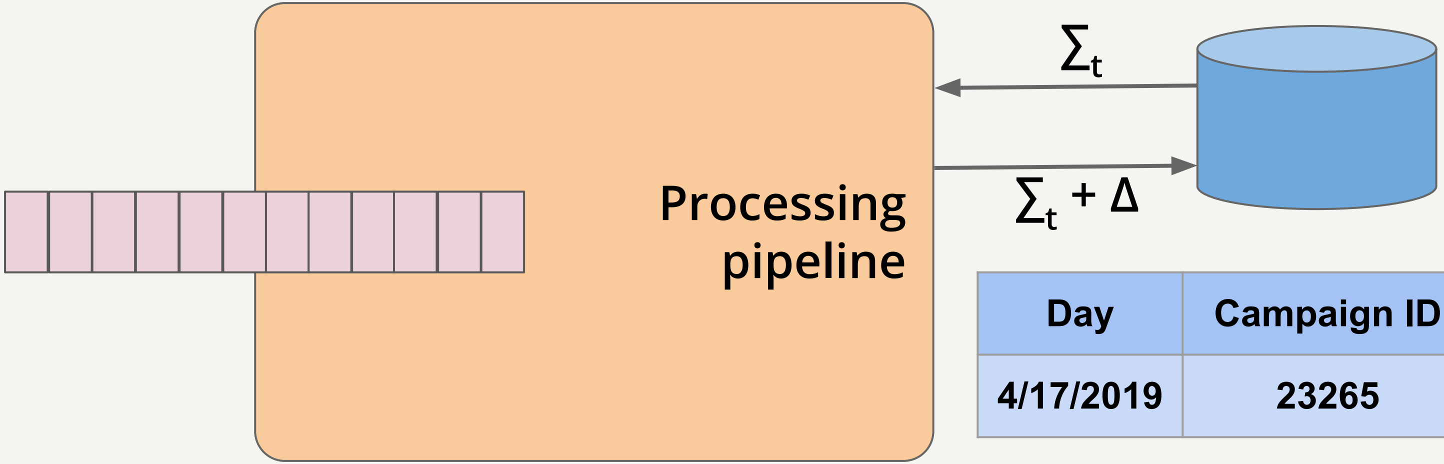
Day	Campaign ID	Number of clicks
4/17/2019	23265	35

- Cassandra has a **Counter** column type
- Integer type with increment and decrement

However...

- **Counter** is not meant to be idempotent
- Good for approximate metrics (likes/follows)
- Reported discrepancies of up to 5%
- Discrepancies due to being distributed
- No plans to make it idempotent

Alright...

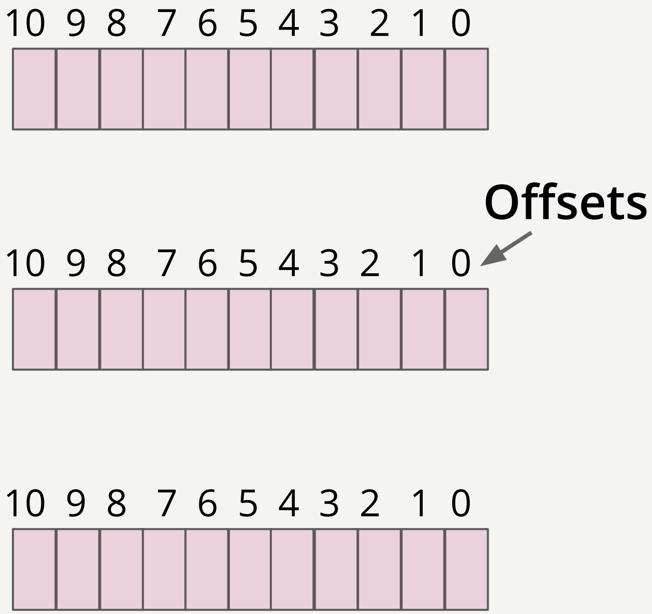


Day	Campaign ID	Number of clicks
4/17/2019	23265	35

- Use Cassandra for the current count
- Increment in Spark and update Cassandra

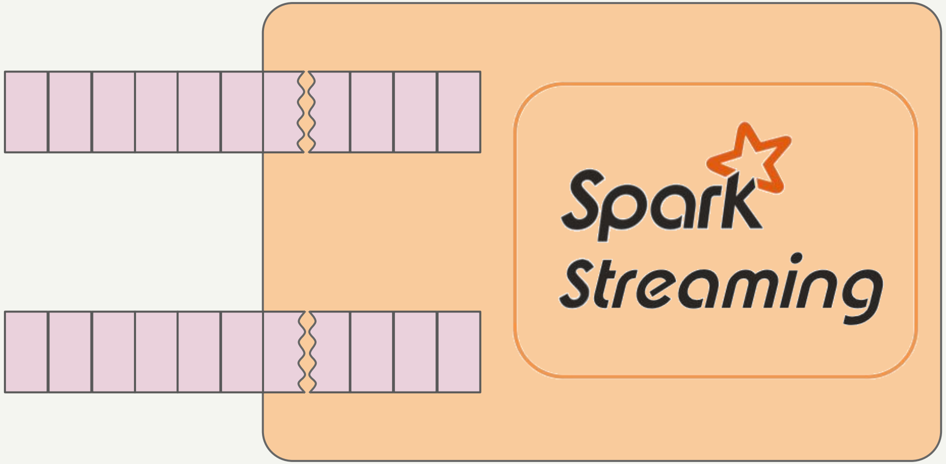
Kafka 101

Partitions



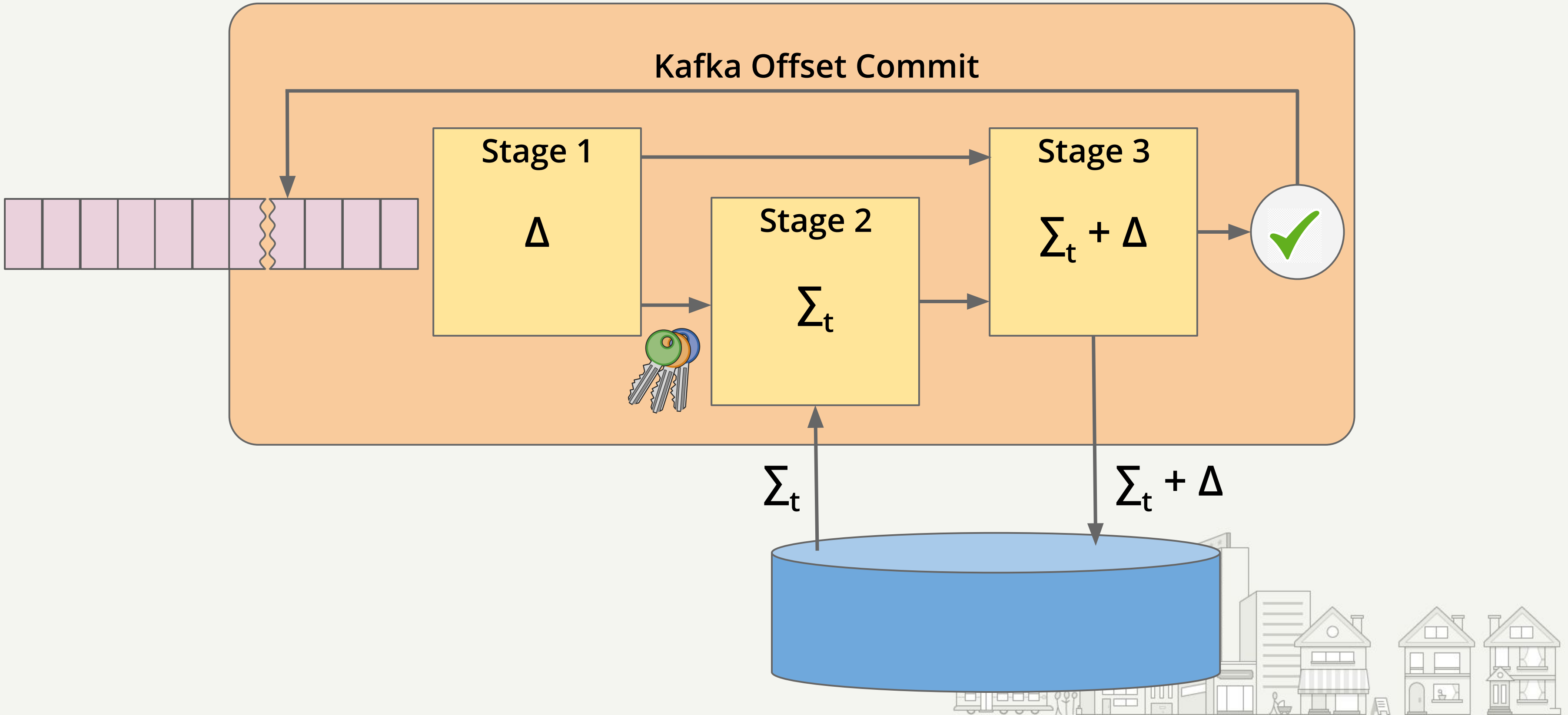
- Data is in partitions
- Partition is ordered
- Consumers track their own progress

Spark Streaming 101



- Micro-batching
- No pipelining
- App manages offset commits

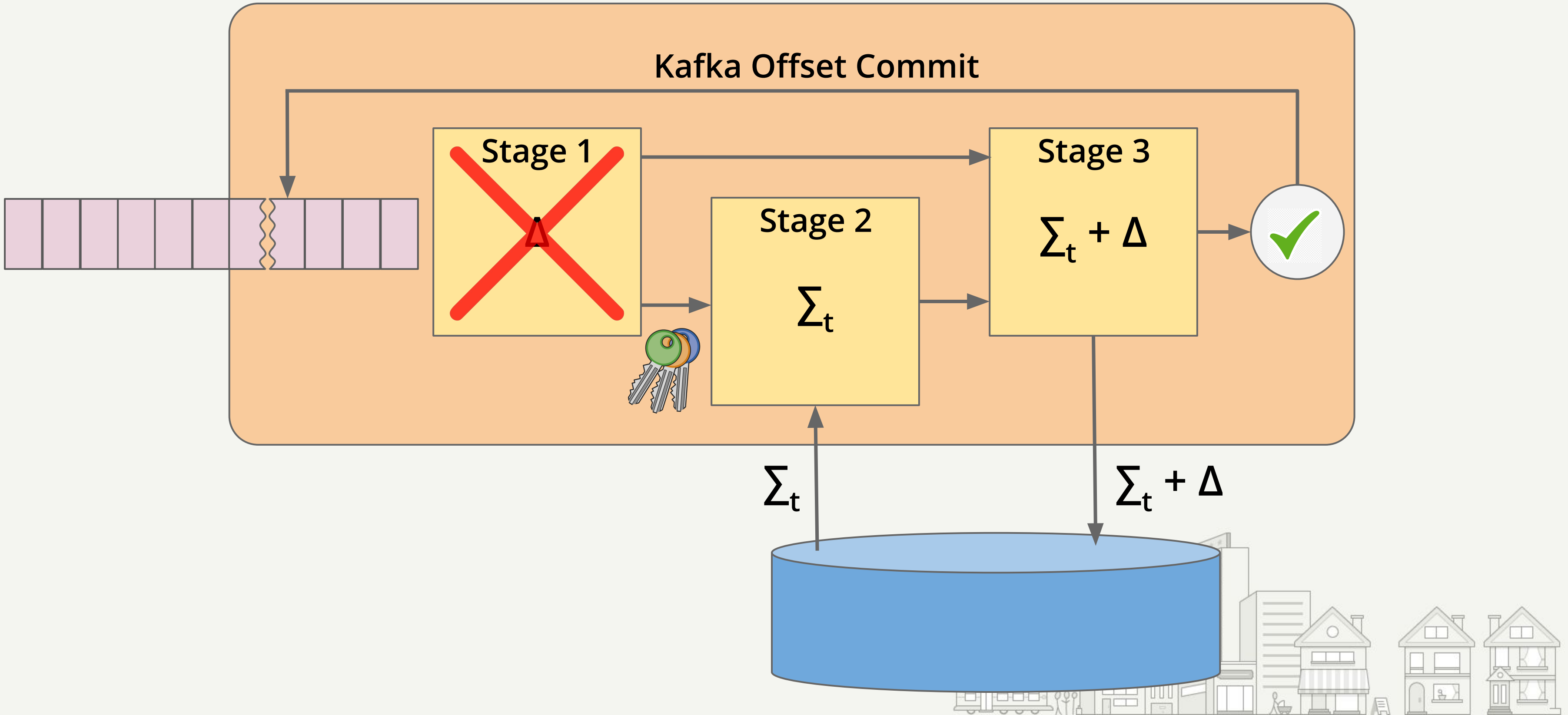
Putting them together



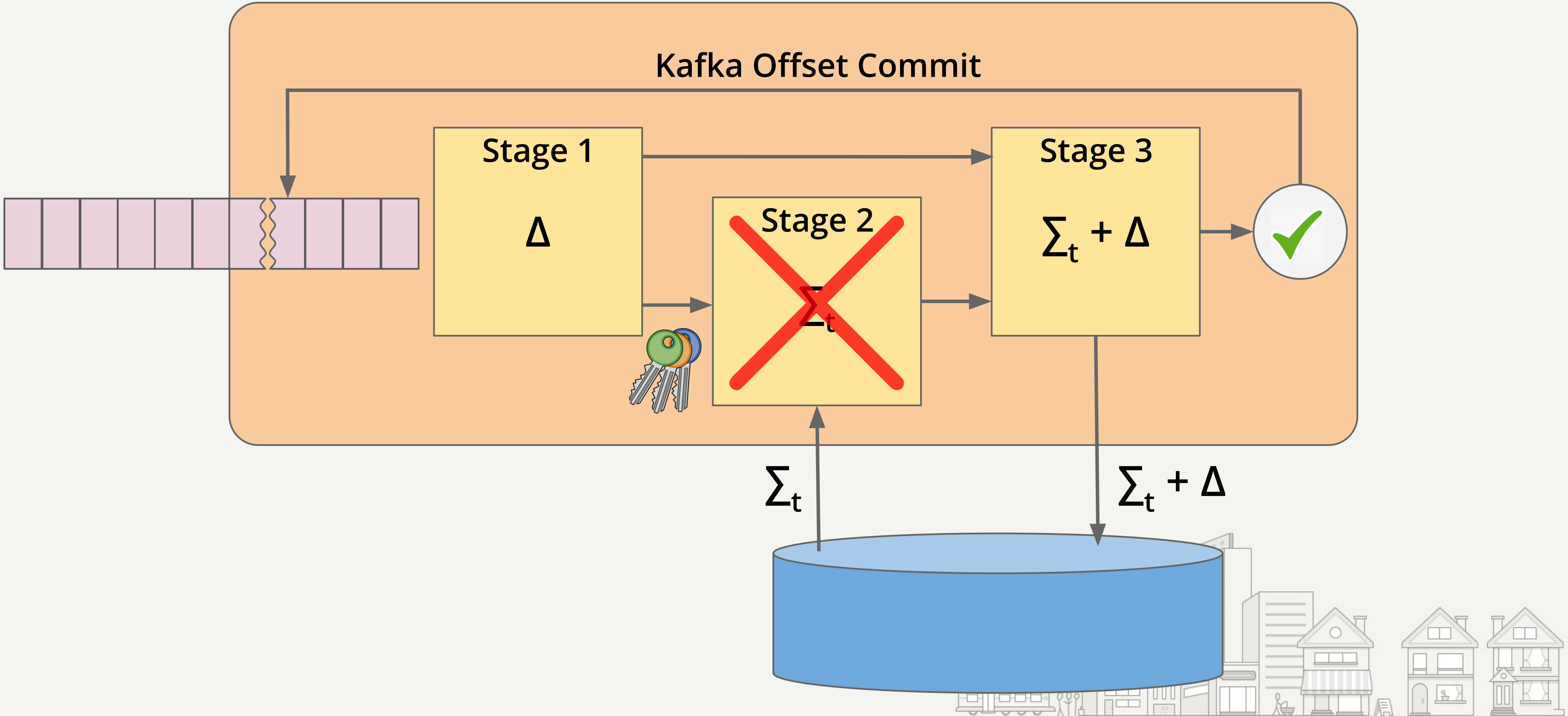
In the words of Ken Arnold

Failure is the defining difference between distributed and local programming, so you have to design distributed systems with the expectation of failure. Imagine asking people, "If the probability of something happening is one in ten to the thirteenth, how often would it happen?" Your natural human sense would be to answer, "Never." That is an infinitely large number in human terms. But if you ask a physicist, she would say, "All the time. In a cubic foot of air, those things happen all the time." When you design distributed systems, you have to say, "Failure happens all the time." So when you design, you design for failure. It is your number one concern.

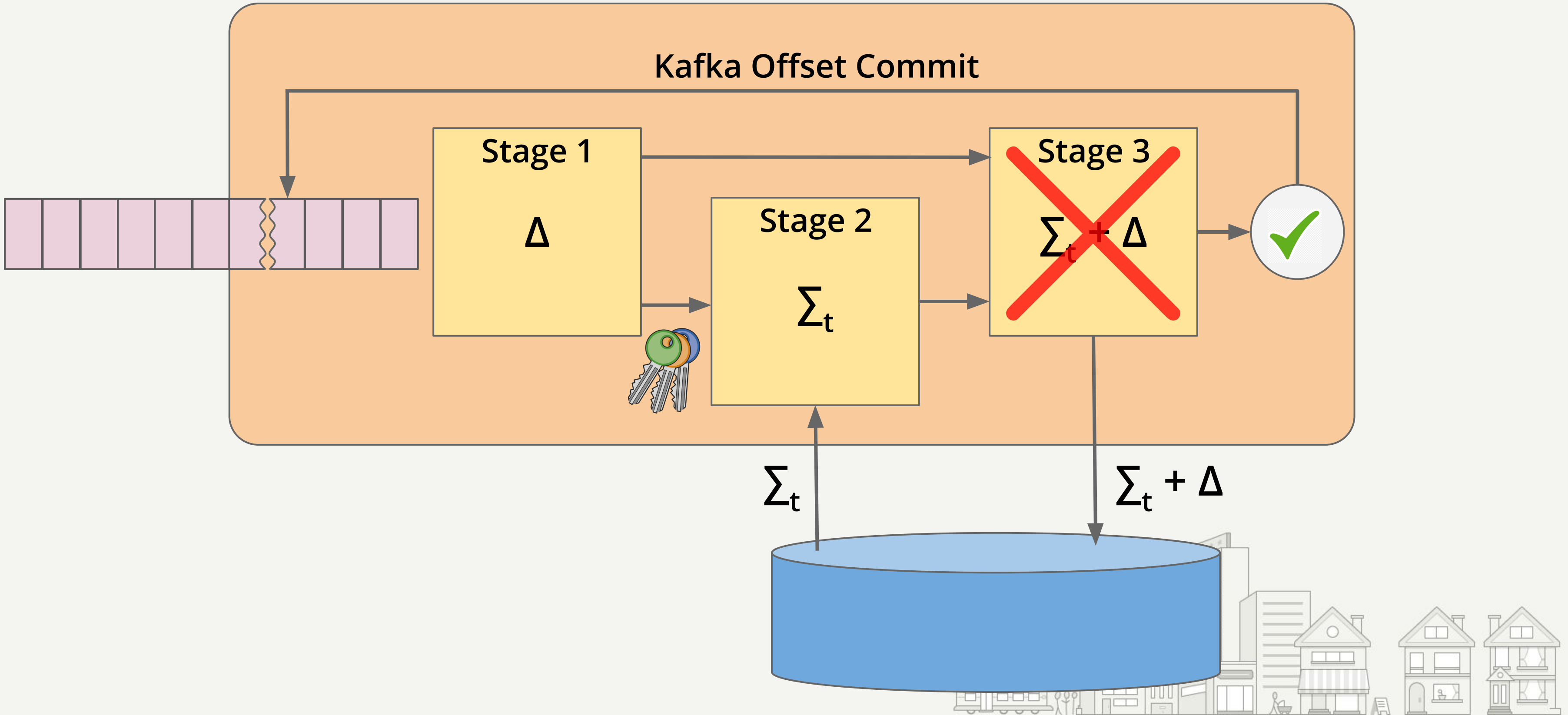
Failure Modes



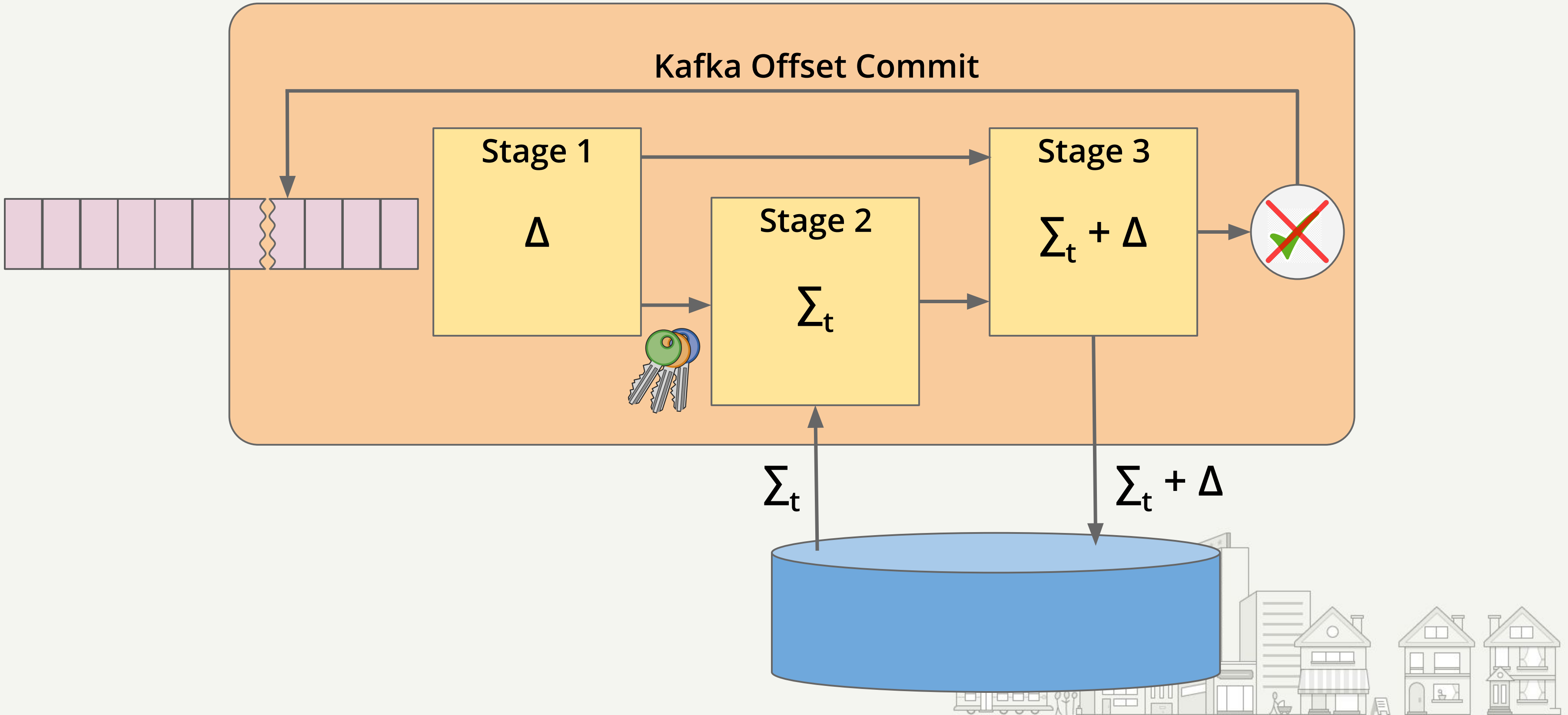
Failure Modes



Failure Modes



Failure Modes

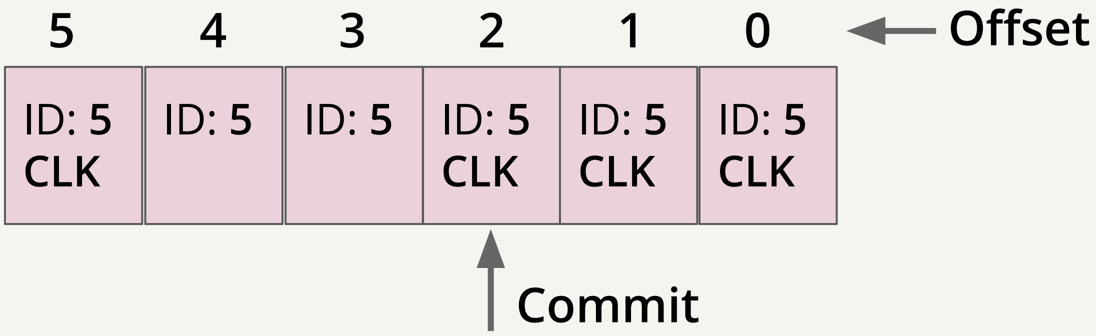


At Least + At Most = Exactly-once

- Should be able to distinguish processed data
- Versioning rows is one way to do it
- Versions need to be monotonically increasing
- Data in Kafka partitions are already ordered
- Versioning can leverage data order

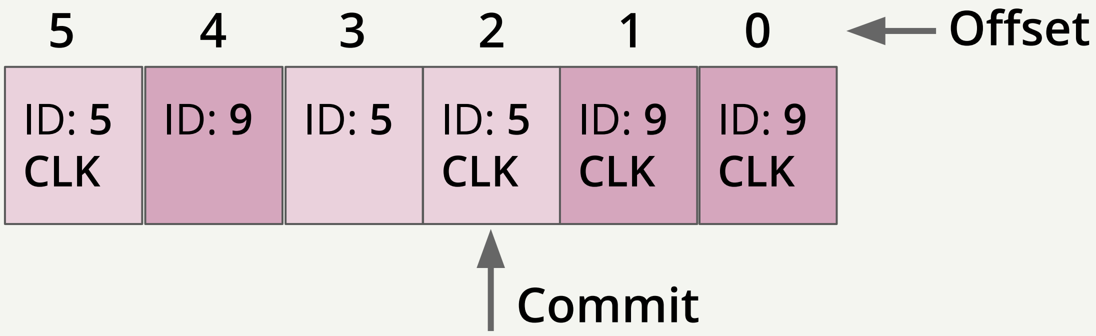
Basic Idea

Day	Campaign ID	Number of clicks	Version
4/17/2019	5	3	2



Basic Idea

Day	Campaign ID	Number of clicks	Version
4/17/2019	5	1	2
4/17/2019	9	2	1



Basic Idea

5	4	3	2	1	0
ID: 5 CLK	ID: 9	ID: 9 CLK	ID: 5	ID: 5 CLK	ID: 9 CLK

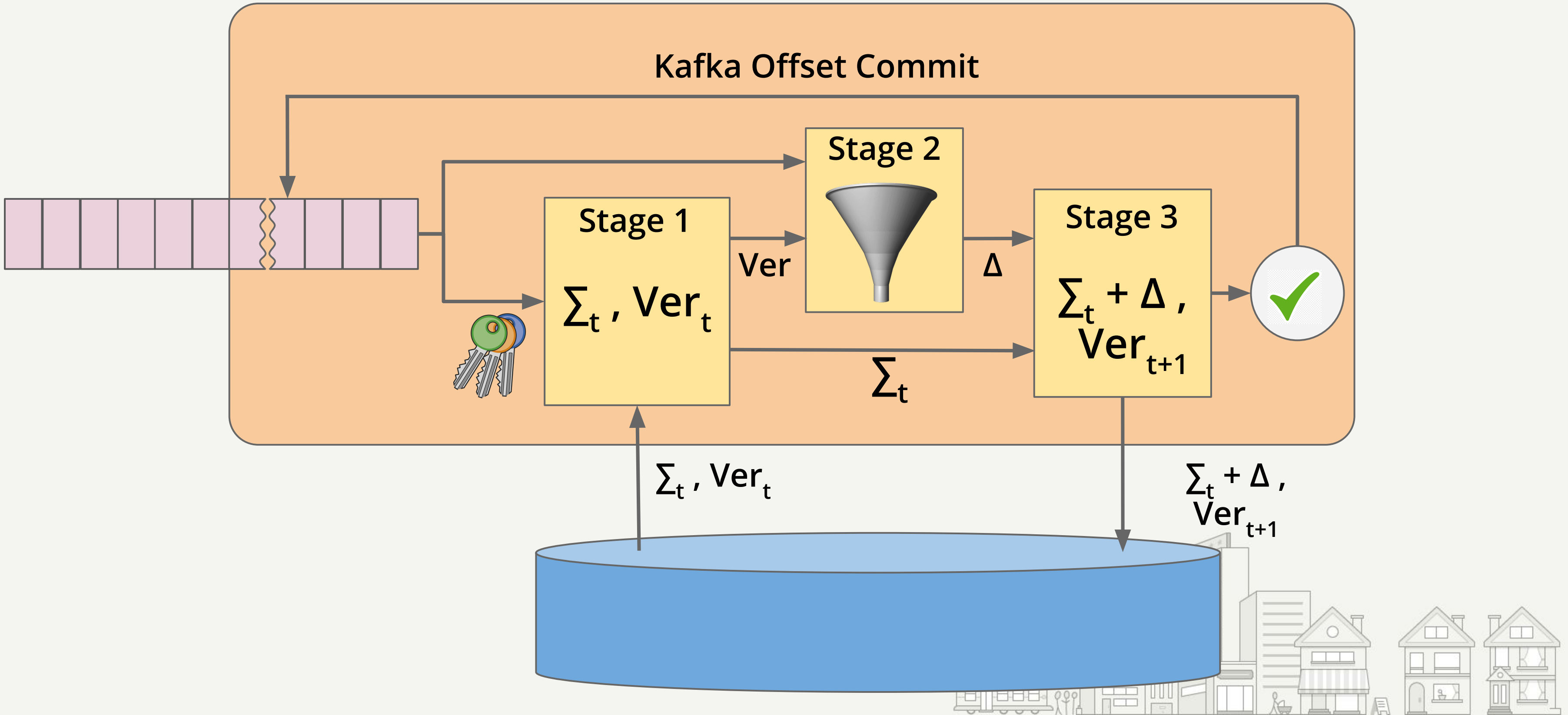
Partition 0

5	4	3	2	1	0
ID: 5 CLK	ID: 9	ID: 5	ID: 5 CLK	ID: 9 CLK	ID: 9 CLK

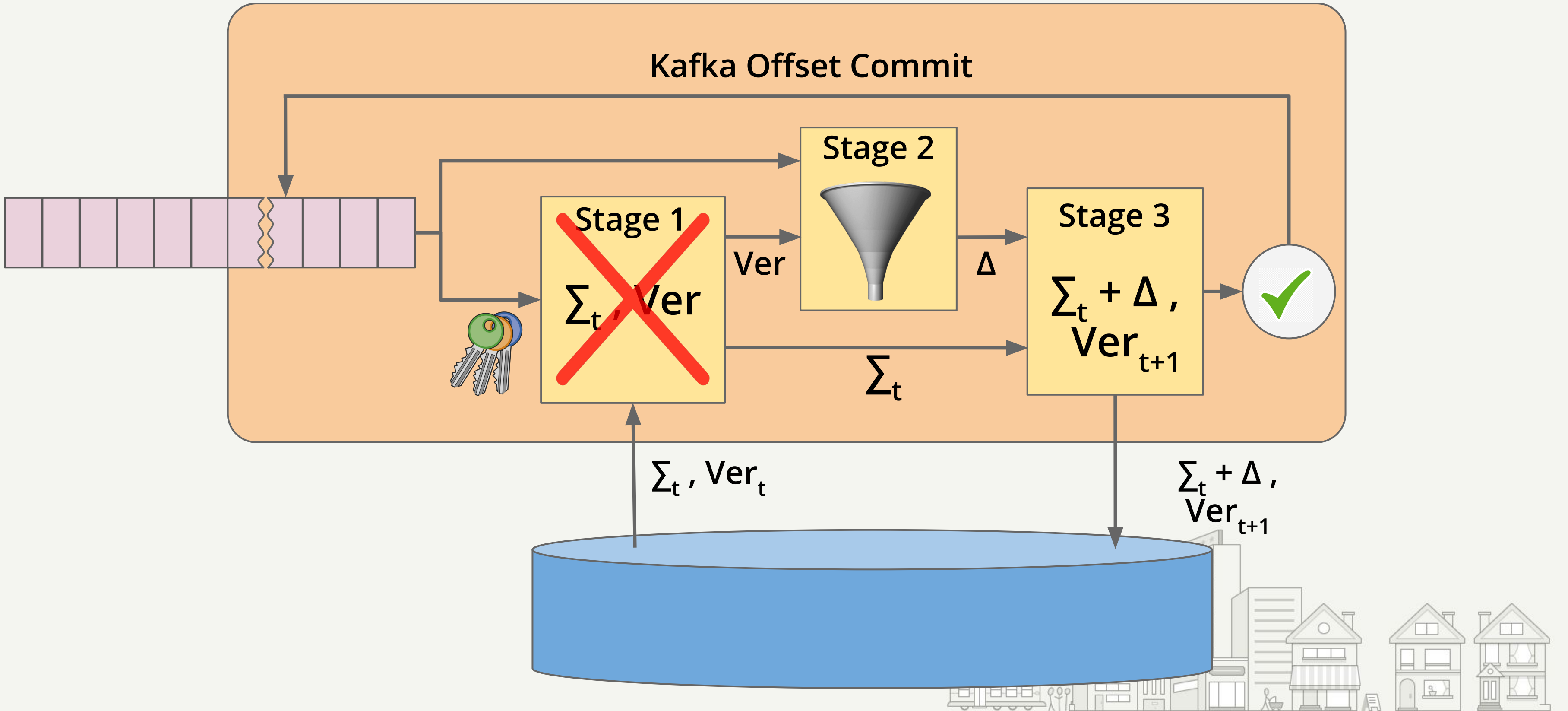
Partition 1

Day	Campaign ID	Number of clicks	Version
4/17/2019	5	2	P0: 2 P1: 3
4/17/2019	9	3	P0: 0 P1: 1

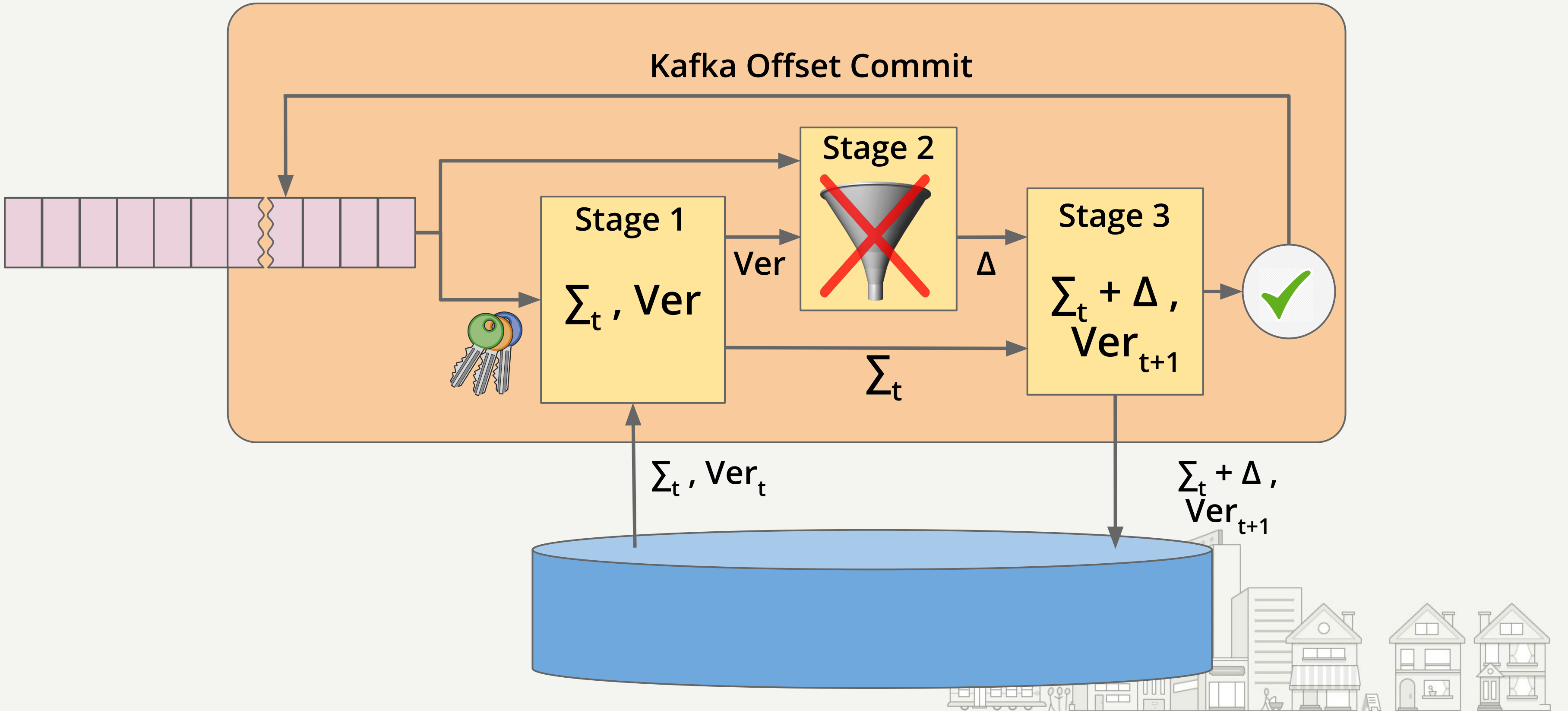
Exactly-once Aggregation



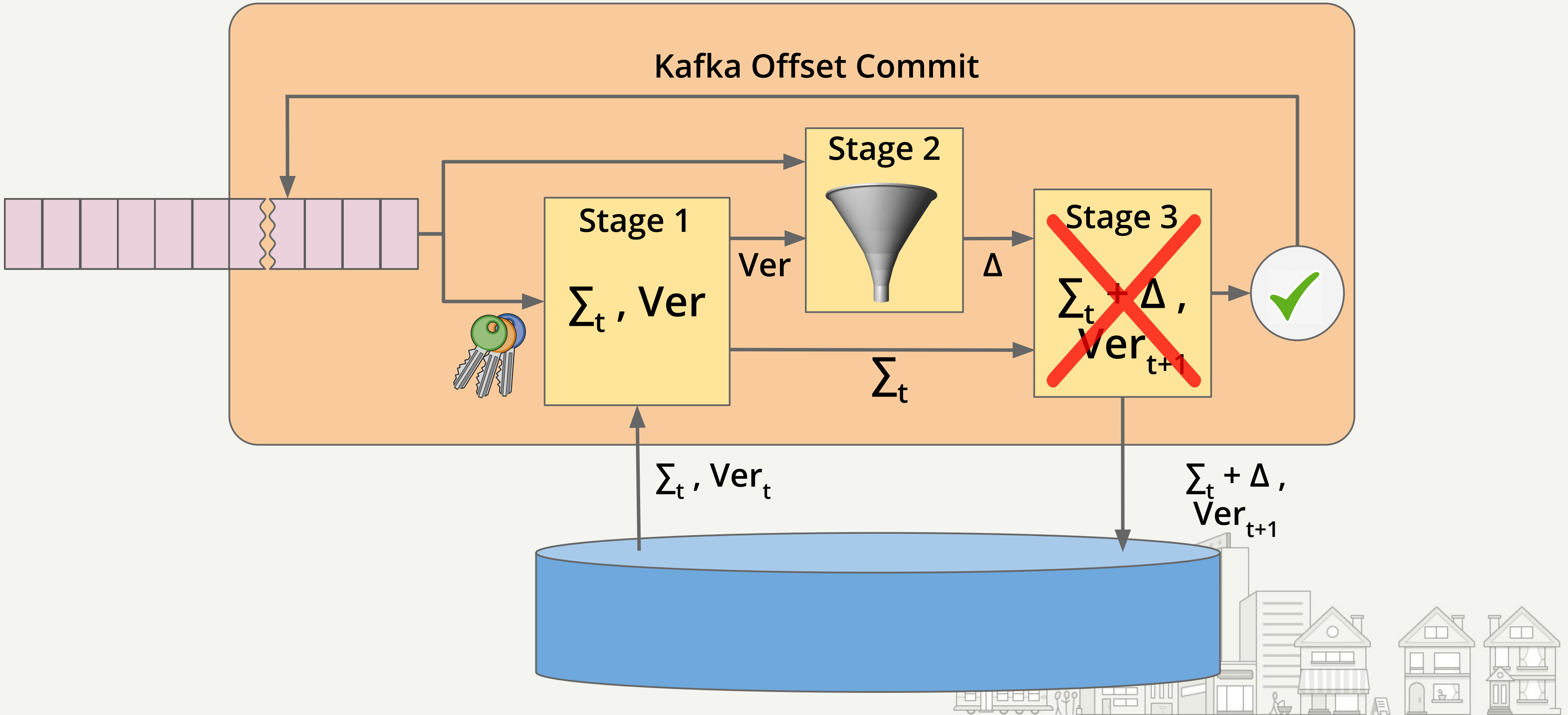
Exactly-once Aggregation



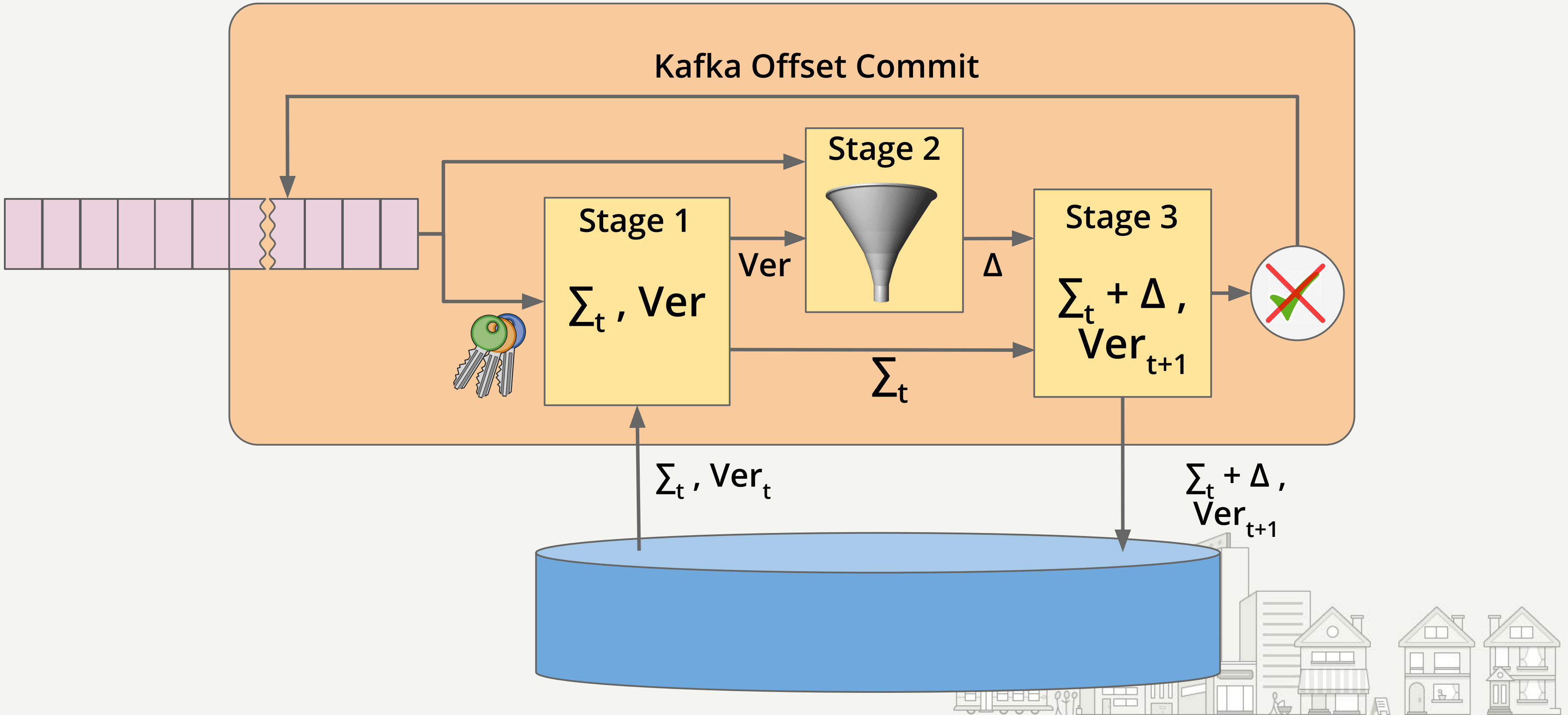
Exactly-once Aggregation



Exactly-once Aggregation



Exactly-once Aggregation



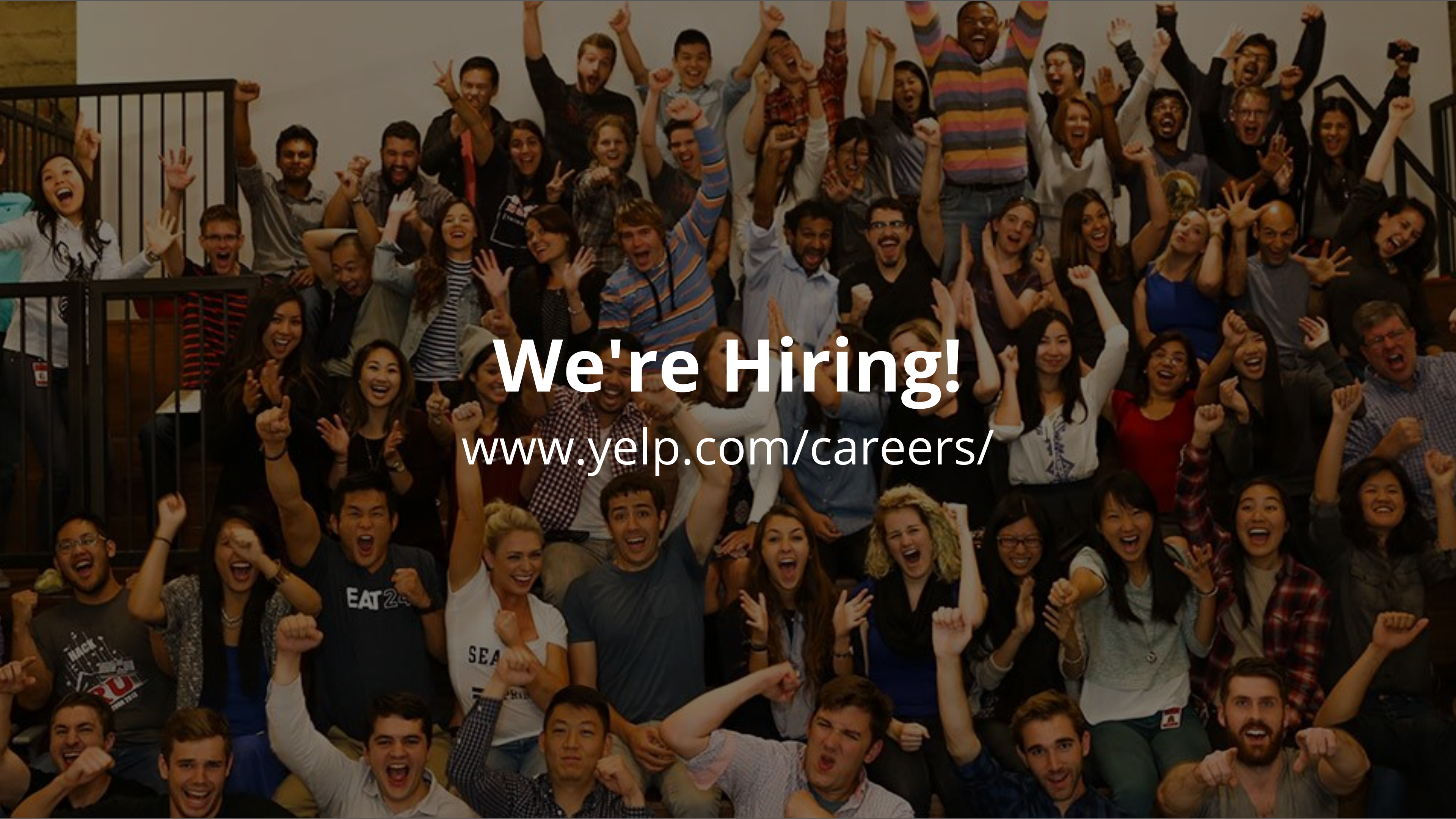
Generalization

```python
def agg_func(partial_agg, new_val):
    new_agg = ...
    return new_agg
```

- Aggregation logic is in the pipeline
- Logic can be arbitrarily complex
- Does not have to be a mathematical function
- Strings, sets, lists, maps, etc.

What’s next?

- Windowed joins
  - As a specialization of aggregation
  - Allows for arbitrary business rules in joins
- Deduplication within aggregation
  - Input streams can typically have duplicates



We're Hiring!

www.yelp.com/careers/

fb.com/YelpEngineers

@YelpEngineering

engineeringblog.yelp.com

github.com/yelp