WHY AND *HOW* TO LEVERAGE THE POWER AND SIMPLICITY OF SQL ON APACHE FLINK®

FABIAN HUESKE, SOFTWARE ENGINEER

dataArtisans

ABOUT ME

- Apache Flink PMC member & ASF member
  - Contributing since day 1 at TU Berlin
  - Focusing on Flink's relational APIs since ~2 years

- Co-author of "Stream Processing with Apache Flink"
  - Work in progress…

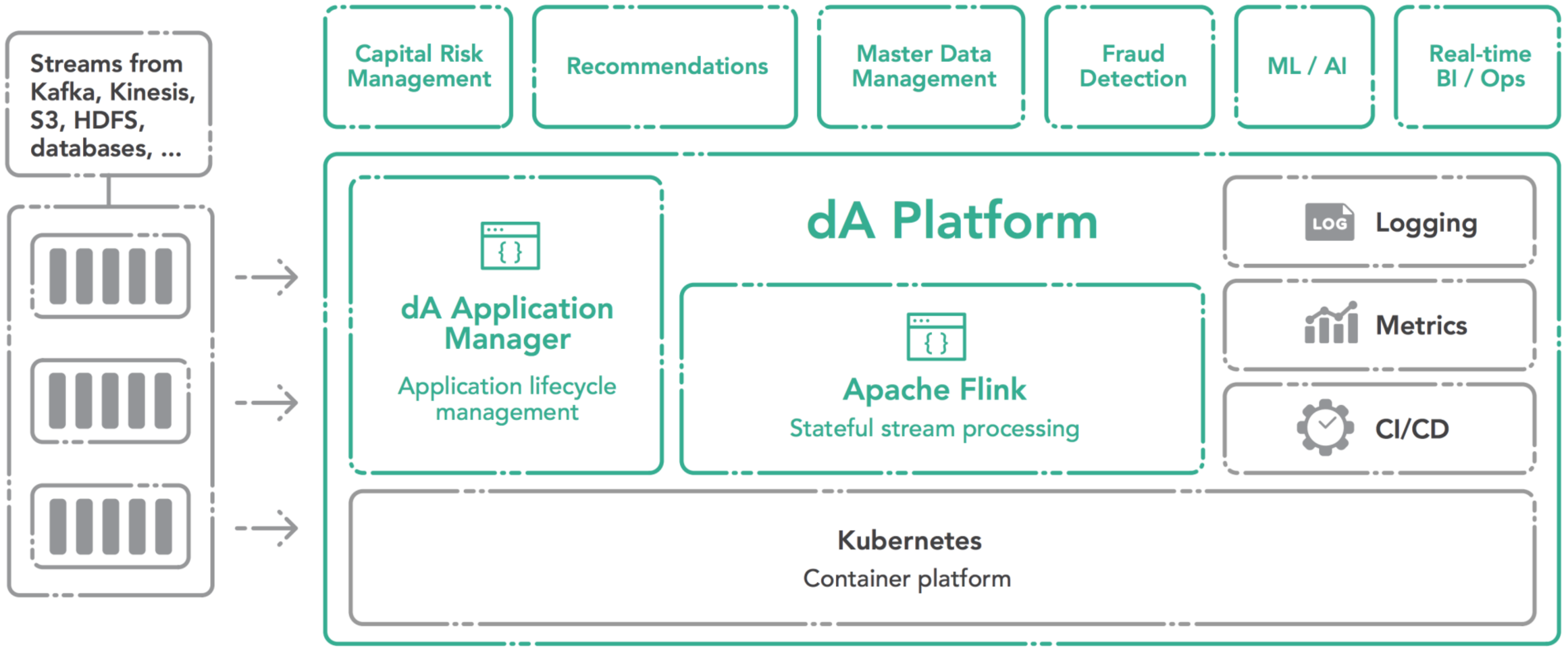
- Co-founder & Software Engineer at data Artisans

ABOUT DATA ARTISANS

Original creators of
Apache Flink®

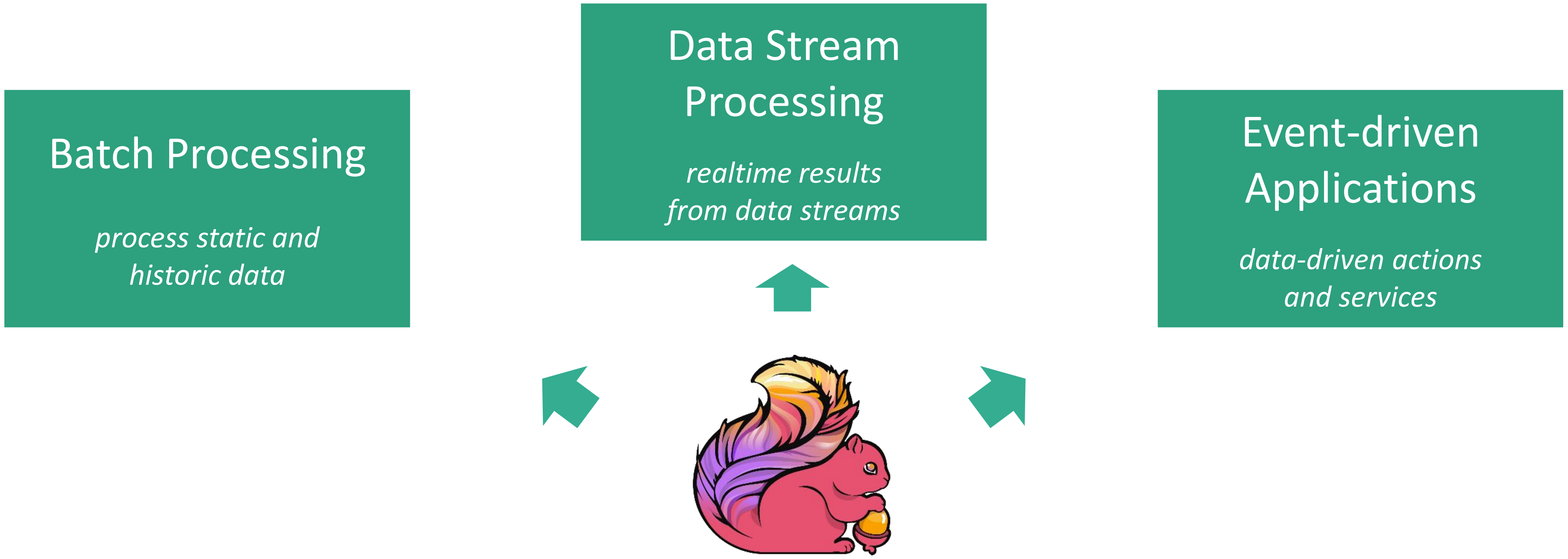
Open Source Apache Flink
+ dA Application Manager

DA PLATFORM



data-artisans.com/download

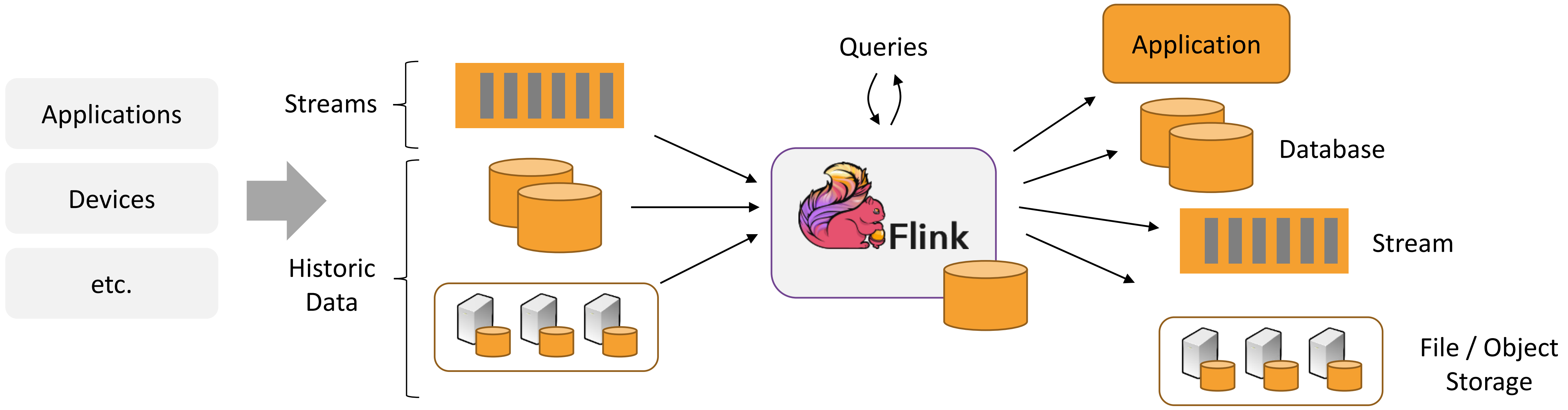
WHAT IS APACHE FLINK?


Batch Processing
process static and historic data
Data Stream Processing
realtime results from data streams
Event-driven Applications
data-driven actions and services

Stateful Computations Over Data Streams

WHAT IS APACHE FLINK?

Stateful computations over streams
real-time and historic
fast, scalable, fault tolerant, in-memory,
event time, large state, exactly-once



POWERED BY APACHE FLINK

Alibaba Group	ING	NETFLIX	UBER	HUAWEI	COMCAST
King	CISCO	accenture	Expedia	ebay	Capital One
DELL EMC	yelp	lyft	here	D TRB	zalando
MediaMath	amadeus	LINE	Telefónica	bouygues TELECOM	Lightbend
bol.com	SK telecom	OVH.com	relayr.	BetterCloud	otto group

FLINK’S POWERFUL ABSTRACTIONS

Layered abstractions to
navigate simple to complex use cases

```
SELECT room, TUMBLE_END(rowtime, INTERVAL '1' HOUR), AVG(temp)
FROM sensors
GROUP BY TUMBLE(rowtime, INTERVAL '1' HOUR), room
```

High-level
Analytics API

SQL / Table API (dynamic tables)

Stream- & Batch
Data Processing

DataStream API (streams, windows)

```
val stats = stream
  .keyBy("sensor")
  .timeWindow(Time.seconds(5))
  .sum((a, b) -> a.add(b))
```

Stateful Event-
Driven Applications

Process Function (events, state, time)

```
def processElement(event: MyEvent, ctx: Context, out: Collector[Result]) = {
  // work with event and state
  (event, state.value) match { … }

  out.collect(…) // emit events
  state.update(…) // modify state

  // schedule a timer callback
  ctx.timerService.registerEventTimeTimer(event.timestamp + 500)
}
```

APACHE FLINK’S RELATIONAL APIS

ANSI SQL

```
SELECT user, COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

LINQ-style Table API

```
tableEnvironment
  .scan("clicks")
  .groupBy('user)
  .select('user, 'url.count as 'cnt)
```

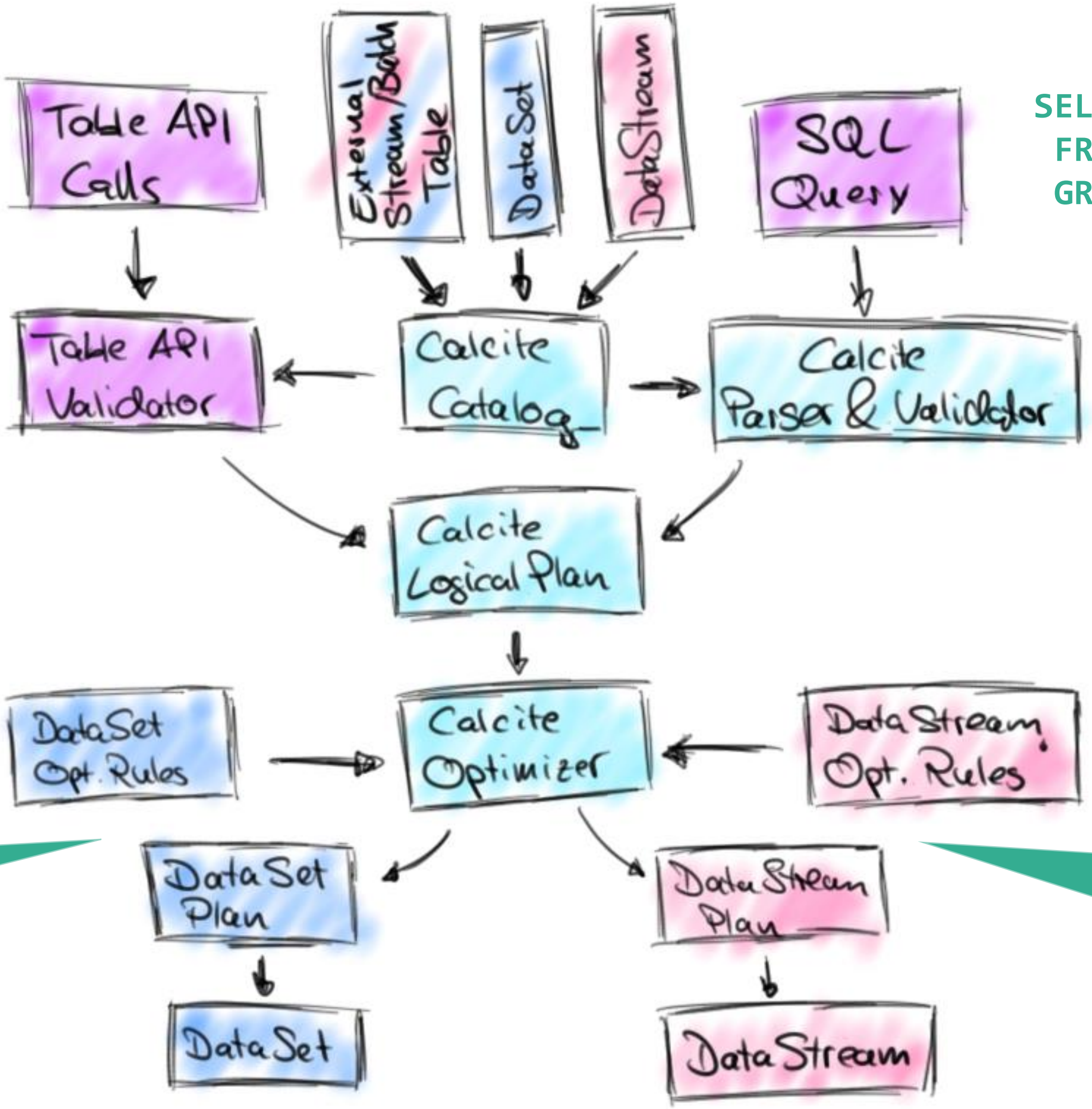
Unified APIs for batch & streaming data

A query specifies exactly the same result regardless whether its input is static batch data or streaming data.

QUERY TRANSLATION

```
tableEnvironment
  .scan("clicks")
  .groupBy('user)
  .select('user, 'url.count as 'cnt)
```

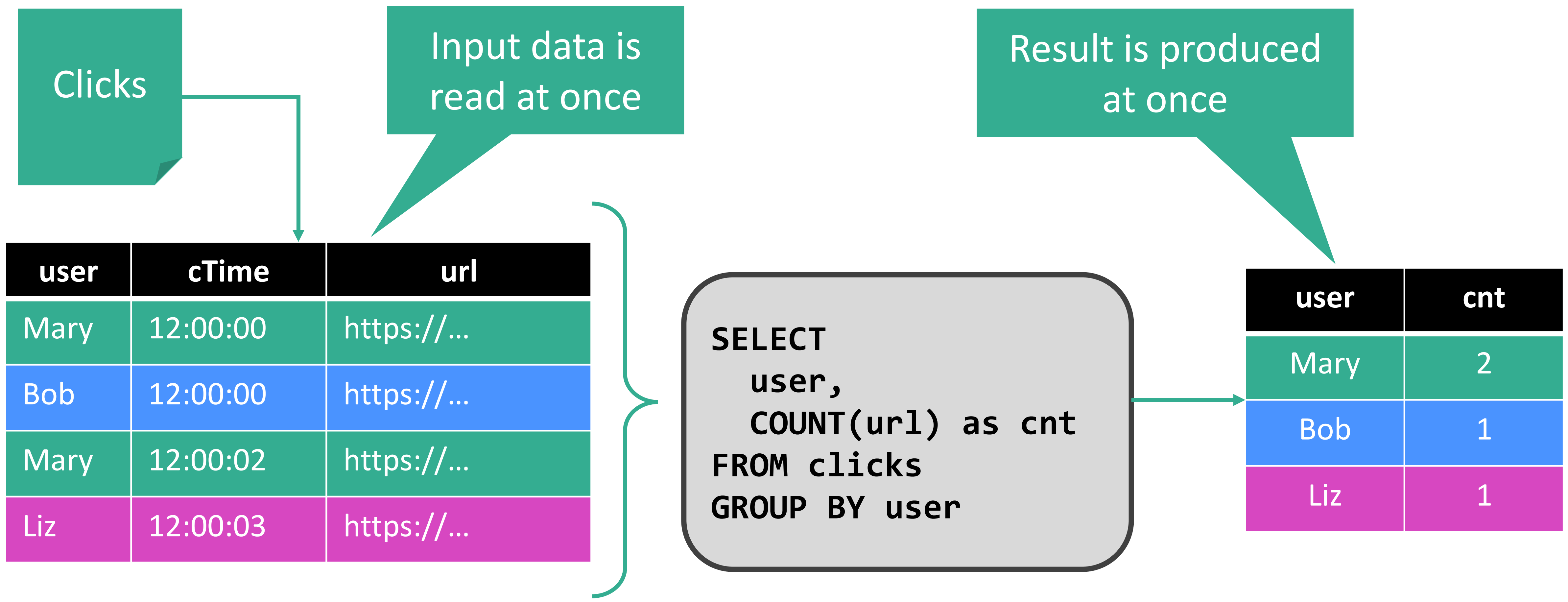
```
SELECT user, COUNT(url) AS cnt
 FROM clicks
 GROUP BY user
```



Input data is **bounded** (batch)

Input data is **unbounded** (streaming)

WHAT IF "CLICKS" IS A FILE?

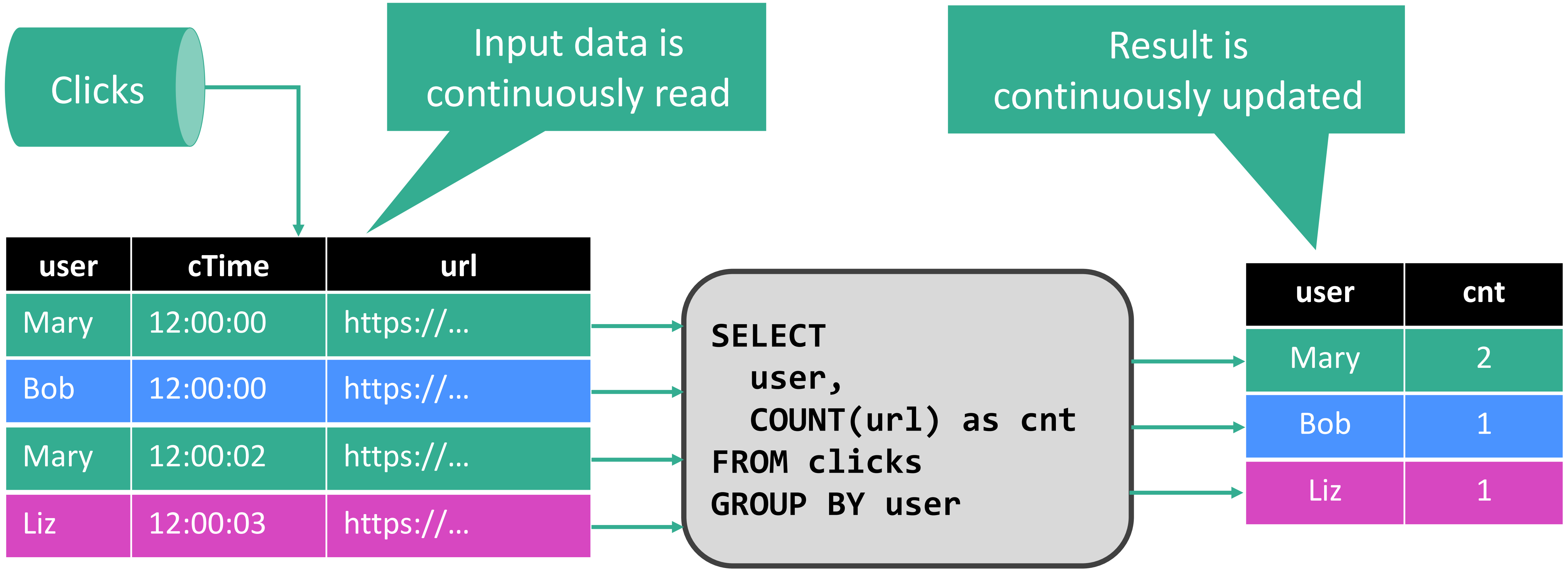


user	cTime	url
Mary	12:00:00	https://…
Bob	12:00:00	https://…
Mary	12:00:02	https://…
Liz	12:00:03	https://…

```
SELECT
  user,
  COUNT(url) as cnt
FROM clicks
GROUP BY user
```

user	cnt
Mary	2
Bob	1
Liz	1

WHAT IF “CLICKS” IS A STREAM?


Clicks
Input data is continuously read
Result is continuously updated

user	cTime	url
Mary	12:00:00	https://…
Bob	12:00:00	https://…
Mary	12:00:02	https://…
Liz	12:00:03	https://…

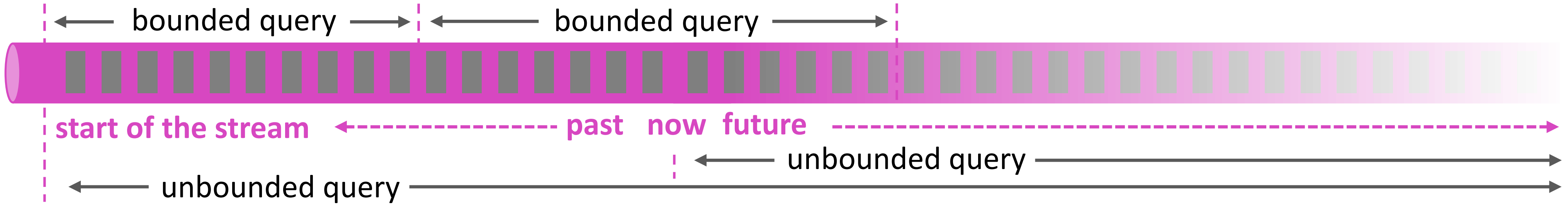
```
SELECT
  user,
  COUNT(url) as cnt
FROM clicks
GROUP BY user
```

user	cnt
Mary	2
Bob	1
Liz	1

The result is the same!

WHY IS STREAM-BATCH UNIFICATION IMPORTANT?

- Usability
  - ANSI SQL syntax: No custom “StreamSQL” syntax.
  - ANSI SQL semantics: No stream-specific results.
- Portability
  - Run the same query on *bounded* and *unbounded* data
  - Run the same query on *recorded* and *real-time* data



- How can we achieve SQL semantics on streams?

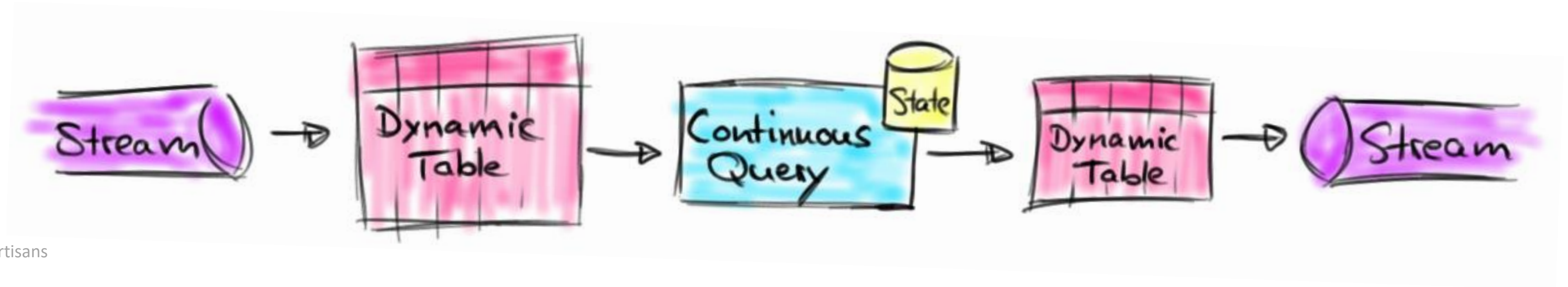
DATABASE SYSTEMS RUN QUERIES ON STREAMS

- Materialized views (MV) are similar to regular views, but persisted to disk or memory
  - Used to speed-up analytical queries
  - MVs need to be updated when the base tables change

- MV maintenance is very similar to SQL on streams
  - Base table updates are a stream of DML statements
  - MV definition query is evaluated on that stream
  - MV is query result and continuously updated

CONTINUOUS QUERIES IN FLINK

- Core concept is a *“Dynamic Table”*
  - Dynamic tables are changing over time
- Queries on dynamic tables
  - produce new dynamic tables (which are updated based on input)
  - do not terminate
- Stream ↔ Dynamic table conversions



STREAM ↔ DYNAMIC TABLE CONVERSIONS

- Append Conversions
  - Records are only inserted/appended

- Upsert Conversions
  - Records are inserted/updated/deleted
  - Records have a (composite) unique key

- Changelog Conversions
  - Records are inserted/updated/deleted

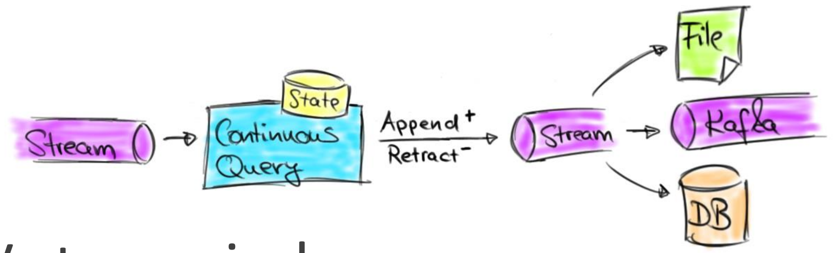
SQL FEATURE SET IN FLINK 1.5.0

- SELECT FROM WHERE
- GROUP BY / HAVING
  - Non-windowed, TUMBLE, HOP, SESSION windows
- JOIN
  - Windowed INNER, LEFT / RIGHT / FULL OUTER JOIN
  - Non-windowed INNER JOIN
- Scalar, aggregation, table-valued UDFs
- SQL CLI Client (beta)

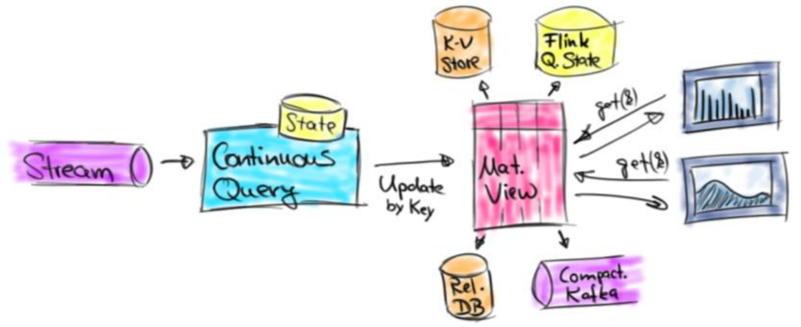
- [streaming only] OVER / WINDOW
  - UNBOUNDED / BOUNDED PRECEDING
- [batch only] UNION / INTERSECT / EXCEPT / IN / ORDER BY

WHAT CAN I BUILD WITH THIS?

- Data Pipelines
  - Transform, aggregate, and move events in real-time
- Low-latency ETL
  - Convert and write streams to file systems, DBMS, K-V stores, indexes, …
  - Ingest appearing files to produce streams



- Stream & Batch Analytics
  - Run analytical queries over bounded and unbounded data
  - Query and compare historic and real-time data
- Power Live Dashboards
  - Compute and update data to visualize in real-time



THE NEW YORK TAXI RIDES DATA SET

- The New York City Taxi & Limousine Commission provides a public data set about past taxi rides in New York City

- We can derive a streaming table from the data

- Table: **`TaxiRides`**

```
rideId:   BIGINT      // ID of the taxi ride
isStart:  BOOLEAN     // flag for pick-up (true) or drop-off (false) event
lon:      DOUBLE      // longitude of pick-up or drop-off location
lat:      DOUBLE      // latitude of pick-up or drop-off location
rowtime:  TIMESTAMP   // time of pick-up or drop-off event
```

IDENTIFY POPULAR PICK-UP / DROP-OFF LOCATIONS

- Compute ***every 5 minutes*** for ***each location*** the ***number of departing and arriving taxis*** of ***the last 15 minutes***.

```
SELECT cell,
  isStart,
  HOP_END(rowtime, INTERVAL '5' MINUTE, INTERVAL '15' MINUTE) AS hopEnd,
  COUNT(*) AS cnt
FROM (SELECT rowtime, isStart, toCellId(lon, lat) AS cell
      FROM TaxiRides)
GROUP BY cell,
  isStart,
  HOP(rowtime, INTERVAL '5' MINUTE, INTERVAL '15' MINUTE)
```

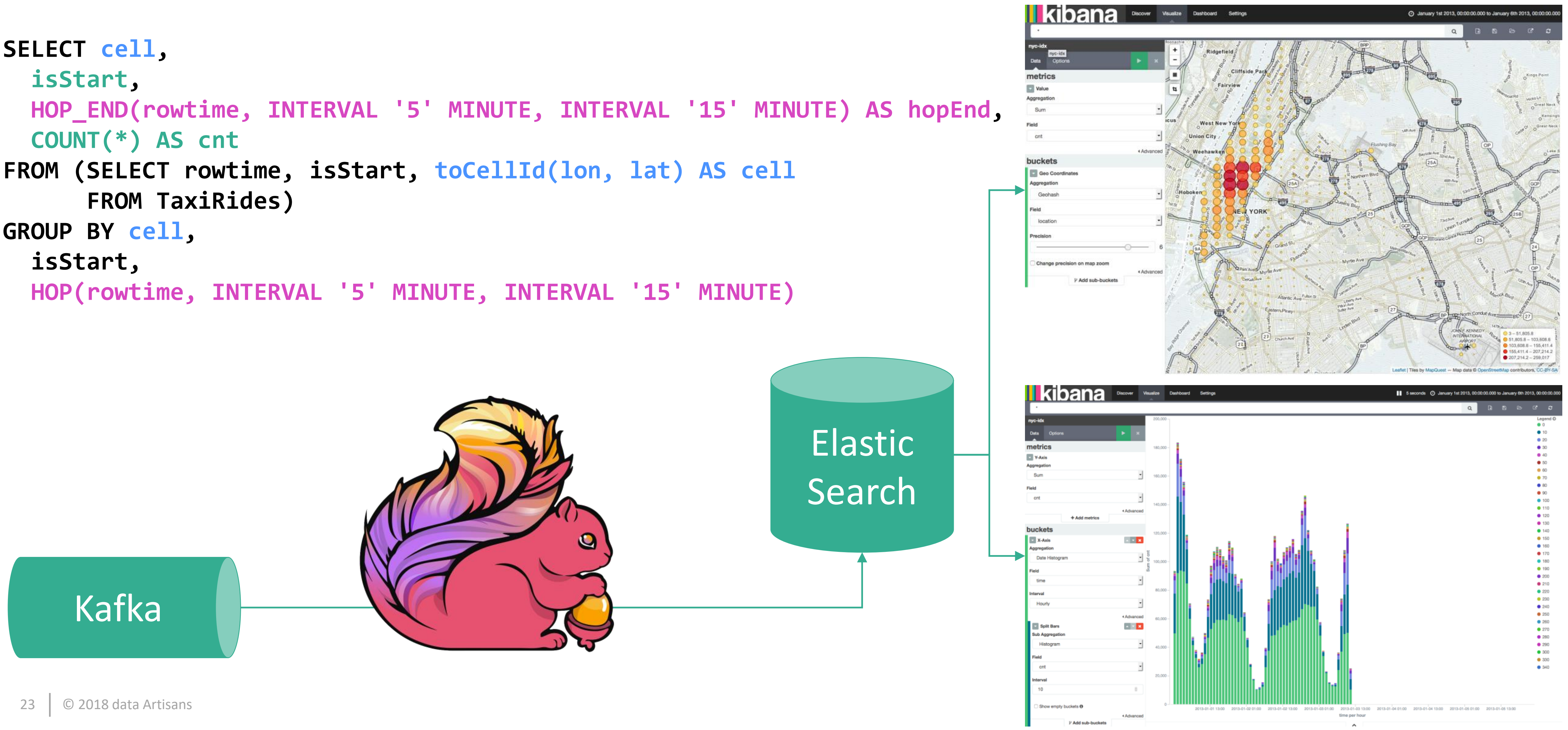
AVERAGE RIDE DURATION PER PICK-UP LOCATION

- ***Join start ride*** and ***end ride*** events ***on rideId*** and compute ***average ride duration per pick-up location***.

```
SELECT pickUpCell,
       AVG(TIMESTAMPDIFF(MINUTE, e.rowtime, s.rowtime) AS avgDuration
FROM (SELECT rideId, rowtime, toCellId(lon, lat) AS pickUpCell
      FROM TaxiRides
      WHERE isStart) s
   JOIN
     (SELECT rideId, rowtime
      FROM TaxiRides
      WHERE NOT isStart) e
    ON s.rideId = e.rideId AND
       e.rowtime BETWEEN s.rowtime AND s.rowtime + INTERVAL '1' HOUR
GROUP BY pickUpCell
```

BUILDING A DASHBOARD

```
SELECT cell,
  isStart,
  HOP_END(rowtime, INTERVAL '5' MINUTE, INTERVAL '15' MINUTE) AS hopEnd,
  COUNT(*) AS cnt
FROM (SELECT rowtime, isStart, toCellId(lon, lat) AS cell
      FROM TaxiRides)
GROUP BY cell,
  isStart,
  HOP(rowtime, INTERVAL '5' MINUTE, INTERVAL '15' MINUTE)
```



SOUNDS GREAT! HOW CAN I USE IT?

- ATM, SQL queries must be embedded in Java/Scala code ☹
  - Tight integration with DataStream and DataSet APIs

- Community focused on internals (until Flink 1.4.0)
  - Operators, types, built-in functions, extensibility (UDFs, extern. catalog)
  - Proven at scale by Alibaba, Huawei, and Uber
  - All built their own submission system & connectors library

- Community neglected user interfaces
  - No query submission client, no CLI
  - No integration with common catalog services
  - Limited set of TableSources and TableSinks

COMING IN FLINK 1.5.0 - SQL CLI

Demo Time!

That’s a nice toy, but …

... can I use it for anything serious?

FLIP-24 – A SQL QUERY SERVICE

- REST service to submit & manage SQL queries
  - `SELECT …`
  - `INSERT INTO SELECT …`
  - `CREATE MATERIALIZE VIEW …`
- Serve results of "SELECT …" queries
- Provide a table catalog (integrated with external catalogs)
- Use cases
  - Data exploration with notebooks like Apache Zeppelin
  - Access to real-time data from applications
  - Easy data routing / ETL from management consoles

CHALLENGE: SERVE DYNAMIC TABLES

Unbounded input yields unbounded results

(Serving bounded results is easy)

```
SELECT user, url
FROM clicks
WHERE url LIKE '%xyz.com'
```

Append-only Table

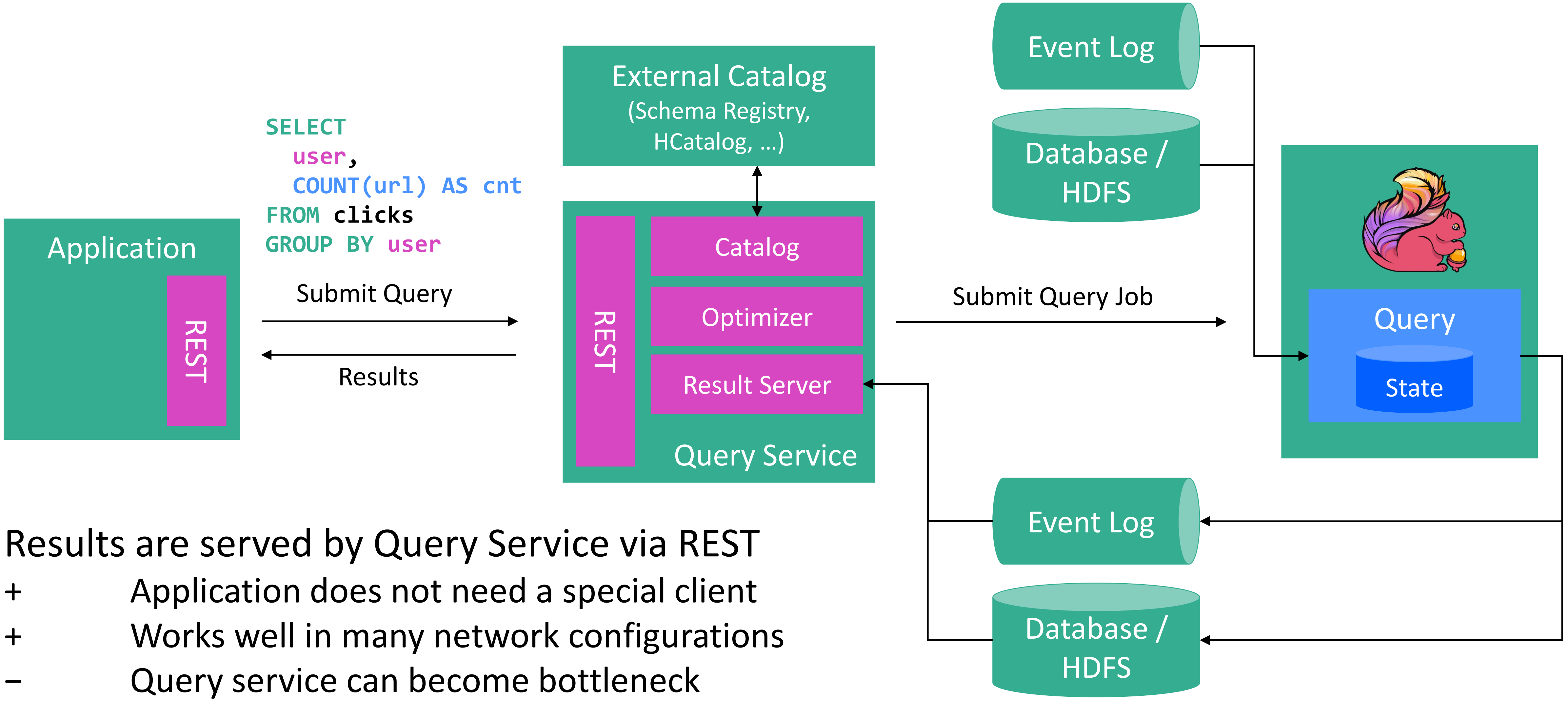
- Result rows are never changed
- Consume, buffer, or drop rows

```
SELECT user, COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

Continuously updating Table

- Result rows can be updated or deleted
- Consume changelog or periodically query result table
- Result table must be maintained somewhere

FLIP-24 – A SQL QUERY SERVICE

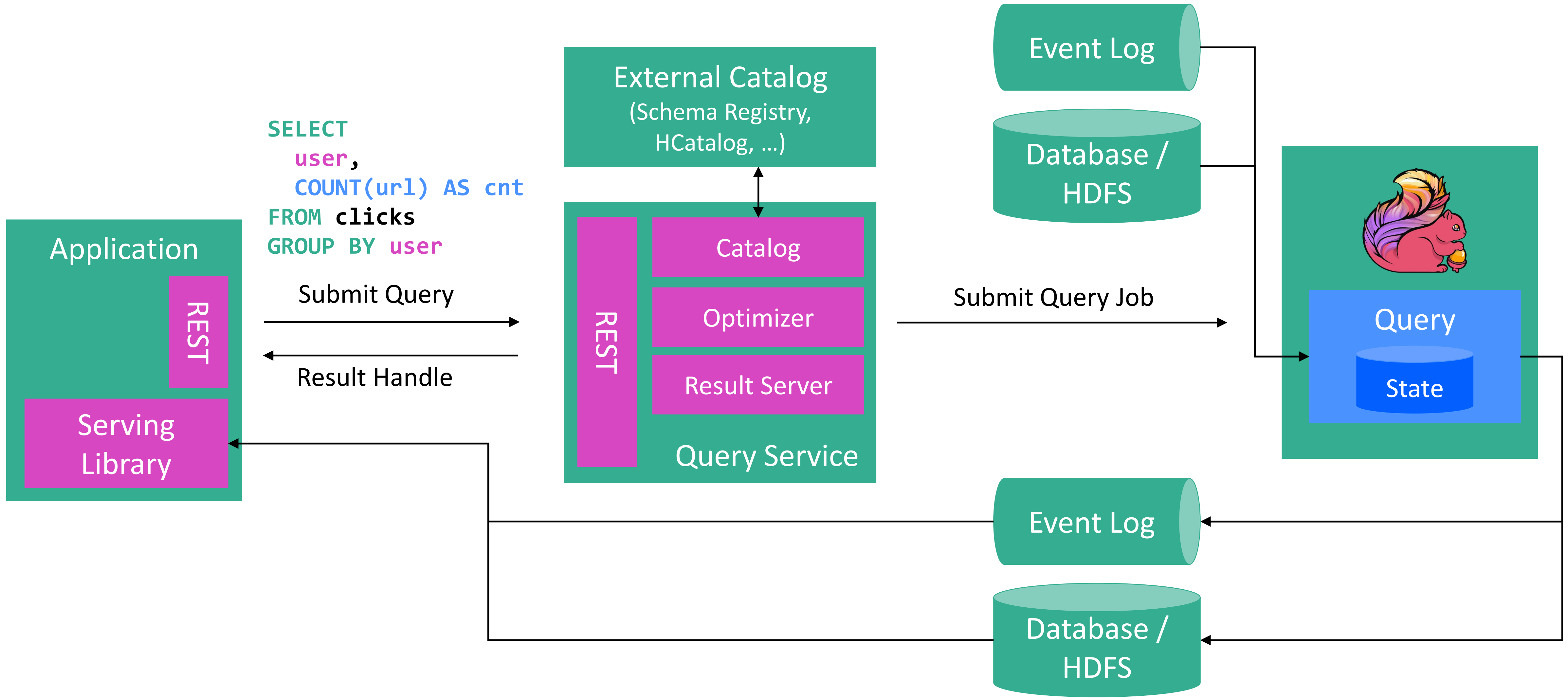


```
SELECT
  user,
  COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

Results are served by Query Service via REST

+ Application does not need a special client
+ Works well in many network configurations
– Query service can become bottleneck

FLIP-24 – A SQL QUERY SERVICE



```
SELECT
  user,
  COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

WE WANT YOUR FEEDBACK!

- The design of SQL Query Service is not final yet.
- Check out FLIP-24 and FLINK-7594
- Share your ideas and feedback and discuss on JIRA or dev@flink.apache.org.

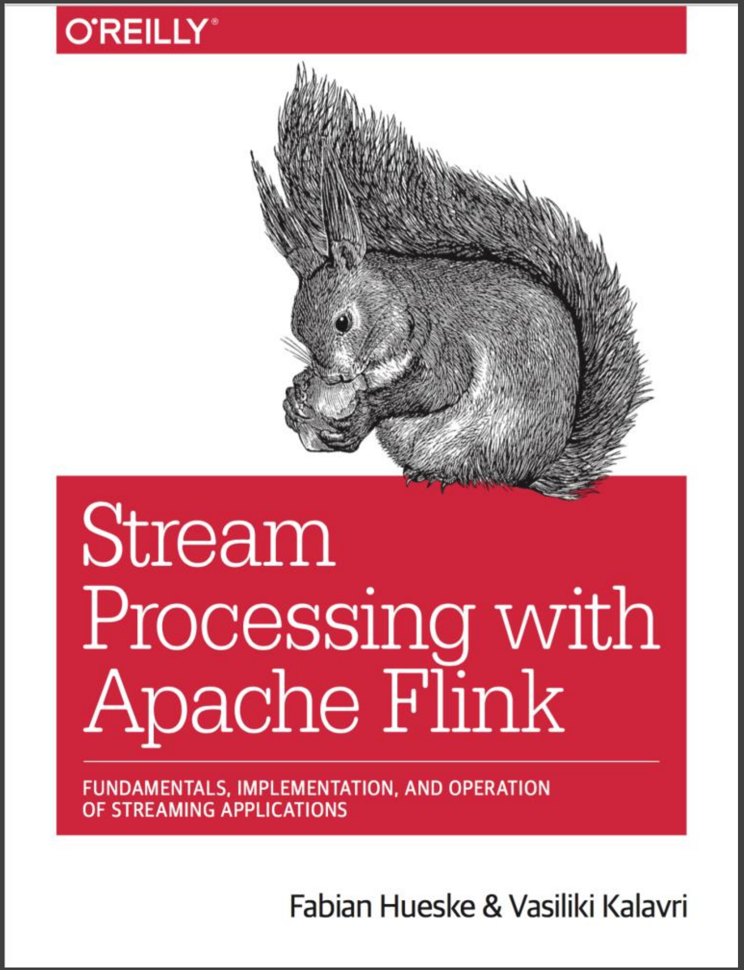
SUMMARY

- Unification of stream and batch is important.

- Flink’s SQL solves many streaming and batch use cases.
- Runs in production at Alibaba, Uber, and others.

- The community is working on improving user interfaces.
- Get involved, discuss, and contribute!

THANK YOU!


O'REILLY

Stream Processing with Apache Flink

FUNDAMENTALS, IMPLEMENTATION, AND OPERATION OF STREAMING APPLICATIONS

Fabian Hueske & Vasiliki Kalavri

Available on O'Reilly Early Release!

dataArtisans

THANK YOU!

@fhueske

@dataArtisans

@ApacheFlink

WE ARE HIRING

data-artisans.com/careers

dataArtisans