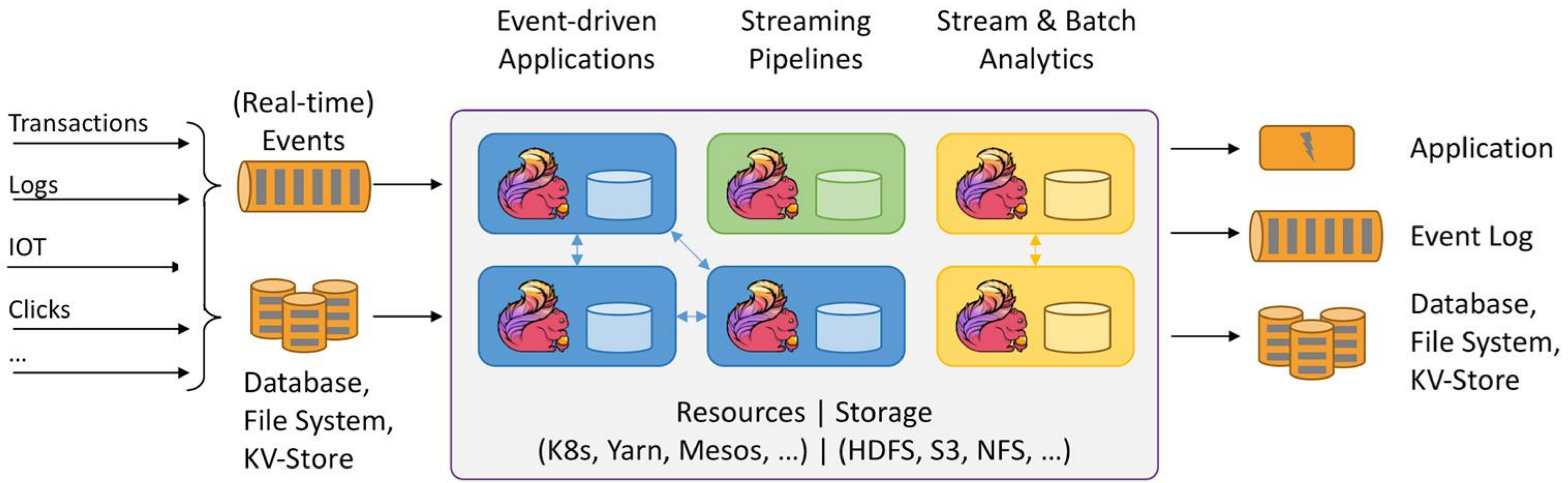
Introduction to SQL on Apache Flink®

Flink SQL Training

https://github.com/ververica/sql-training

Apache Flink is a Distributed Data Processing System


Event-driven Applications
Streaming Pipelines
Stream & Batch Analytics
Transactions
Logs
IOT
Clicks
...
(Real-time) Events
Database, File System, KV-Store
Resources | Storage
(K8s, Yarn, Mesos, ...) | (HDFS, S3, NFS, ...)
Application
Event Log
Database, File System, KV-Store

Scalable and Consistent Data Processing

- Flexible and expressive APIs

- Guaranteed correctness
  - Exactly-once state consistency
  - Event-time semantics

- In-memory processing at massive scale
  - Runs on 10000s of cores
  - Manages 10s TBs of state

Powered By Apache Flink

Alibaba.com aws BetterCloud bouygues TELECOM CapitalOne COMCAST criteo DiDi

D TRB ebay ERICSSON gojek HUAWEI king KLAVIYO 快手

lyft MediaMath MUX oppo otto group OVH.com Razorpay ResearchGate

SK telecom Telefónica NEXT Tencent 腾讯 UBER mi 小米 xiaomi.com yelp zalando

Details about their use cases and more users are listed on Flink's website at https://flink.apache.org/poweredby.html
Also check out the Flink Forward YouTube channel with more than 350 recorded talks at https://www.youtube.com/channel/UCY8_lgiZLZErZPF47a2hXMA

Why SQL for Stream Processing?

- Implementing Flink stream processing apps requires special skills
    - Java/Scala experience
    - In-depth knowledge of streaming concepts like time and state
    - Knowledge of distributed data processing
- Everybody knows and uses SQL
- SQL queries are optimized and efficiently executed
- Unified syntax and semantics for batch & streaming data

Flink SQL in a Nutshell

A standard-compliant SQL service

to query static and streaming data alike

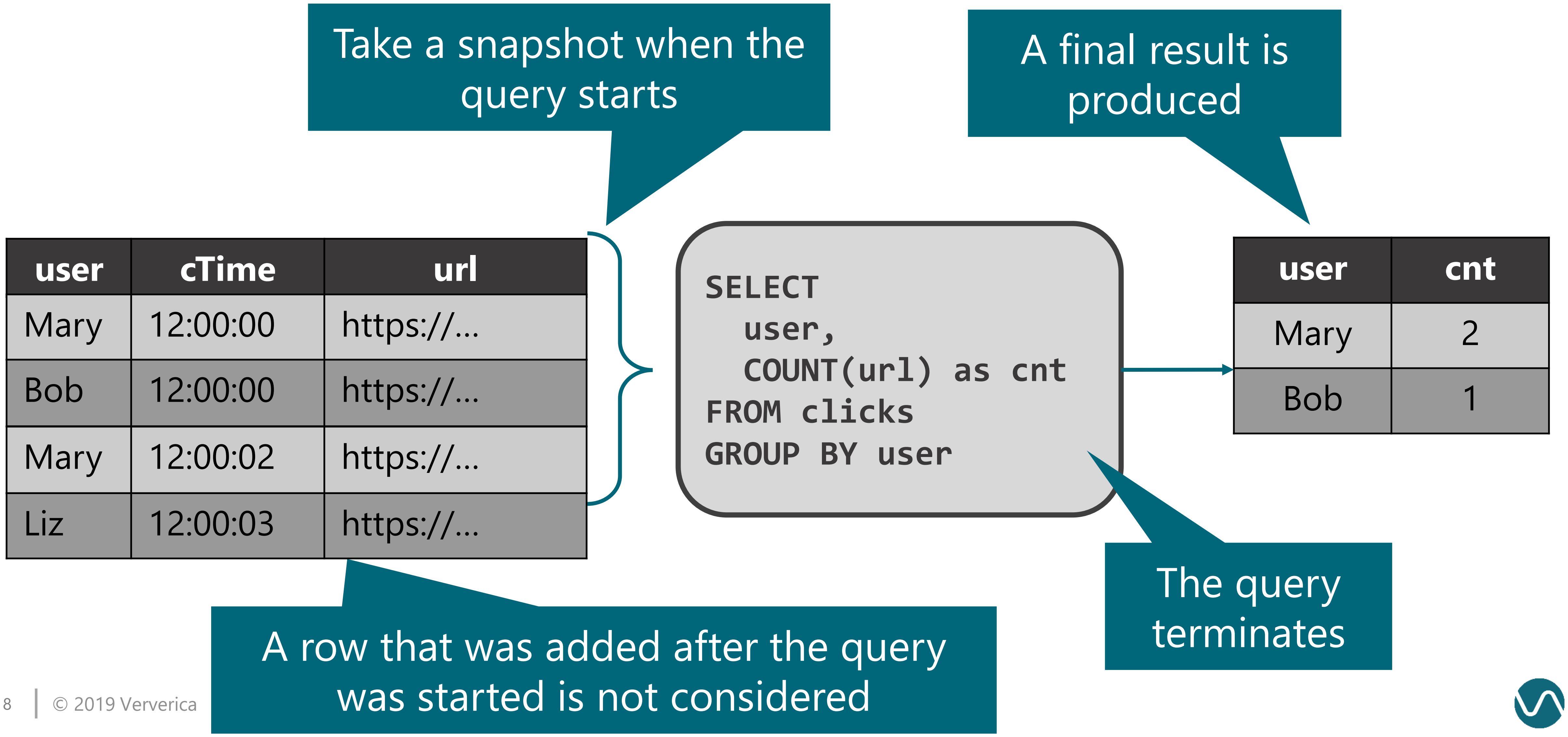
that leverages the performance, scalability, and consistency of

Apache Flink.

How is streaming SQL different from traditional SQL?

- Basically all tables that are processed with SQL queries change over time
    - Transactions from applications
    - Bulk inserts from ETL processes
- Traditional processors run SQL queries on static snapshots of the tables
    - The query input is finite
    - The query result is final and finite
- Stream SQL processors run continuous queries on changing (dynamic) tables
    - The query input is unbounded
    - The query result is never final, continuously updated, and potentially unbounded
- The semantics of a query are the same regardless whether it is executed one-time on a table snapshot or continuously on a changing table

Running a One-time Query on a Changing Table



Take a snapshot when the query starts

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

A row that was added after the query was started is not considered

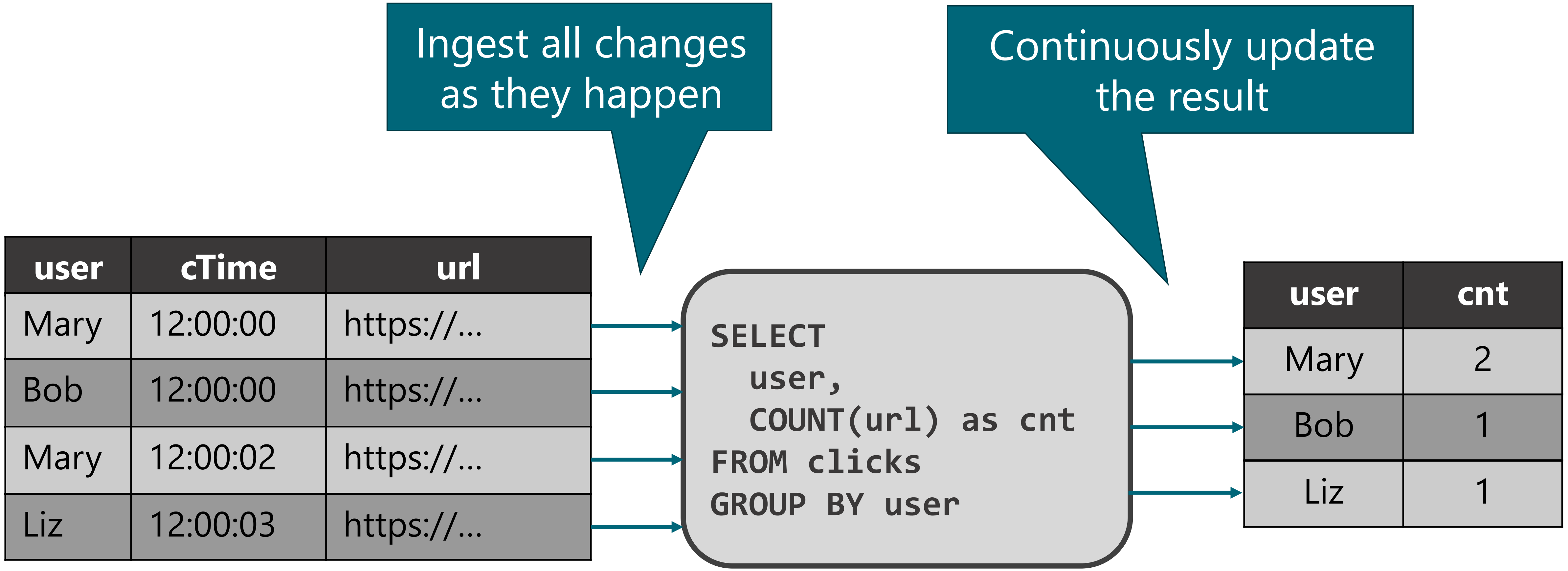
```
SELECT
  user,
  COUNT(url) as cnt
FROM clicks
GROUP BY user
```

The query terminates

A final result is produced

user	cnt
Mary	2
Bob	1

Running a Continuous Query on a Changing Table


Ingest all changes as they happen
Continuously update the result

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 identical to the one-time query (at this point)

SQL Feature Set in Flink 1.10

STREAMING & BATCH

- `SELECT FROM WHERE`
- `GROUP BY [HAVING]`
  - Non-windowed
  - `TUMBLE, HOP, SESSION` windows
- `JOIN`
  - Time-Windowed `INNER + OUTER JOIN`
  - Non-windowed `INNER + OUTER JOIN`
- User-Defined Functions
  - Scalar
  - Aggregation
  - Table-valued

STREAMING ONLY

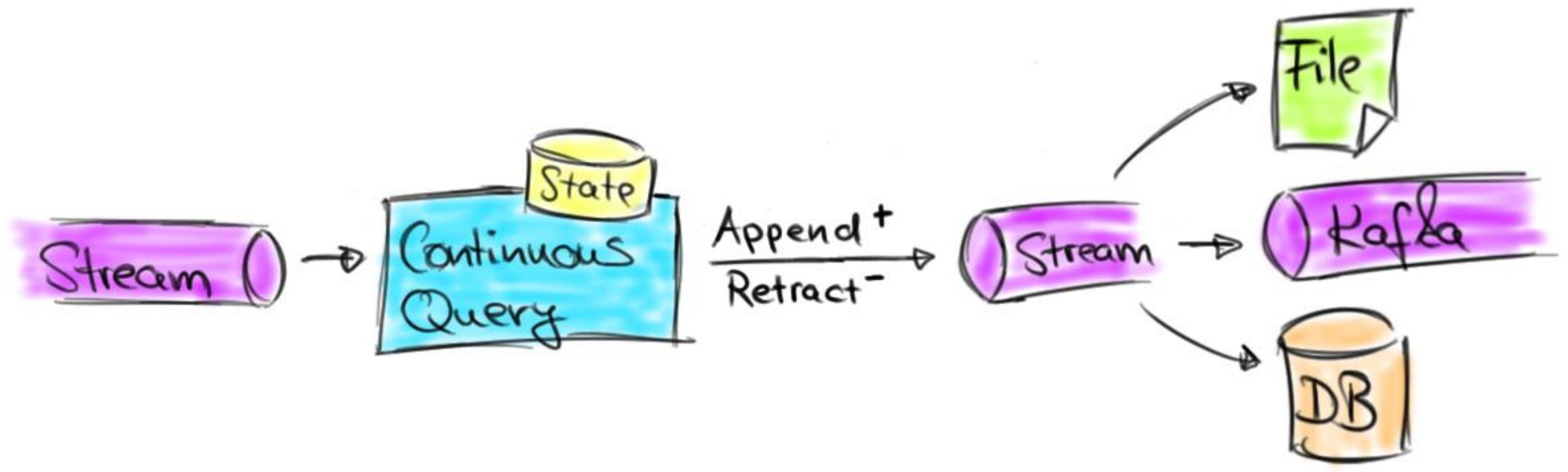
- `OVER / WINDOW`
  - `UNBOUNDED + BOUNDED PRECEDING`
- `INNER JOIN` with
  - Time-versioned table
  - External lookup table
- `MATCH_RECOGNIZE`
  - Pattern Matching/CEP (SQL:2016)

BATCH ONLY

- Full TPC-DS support

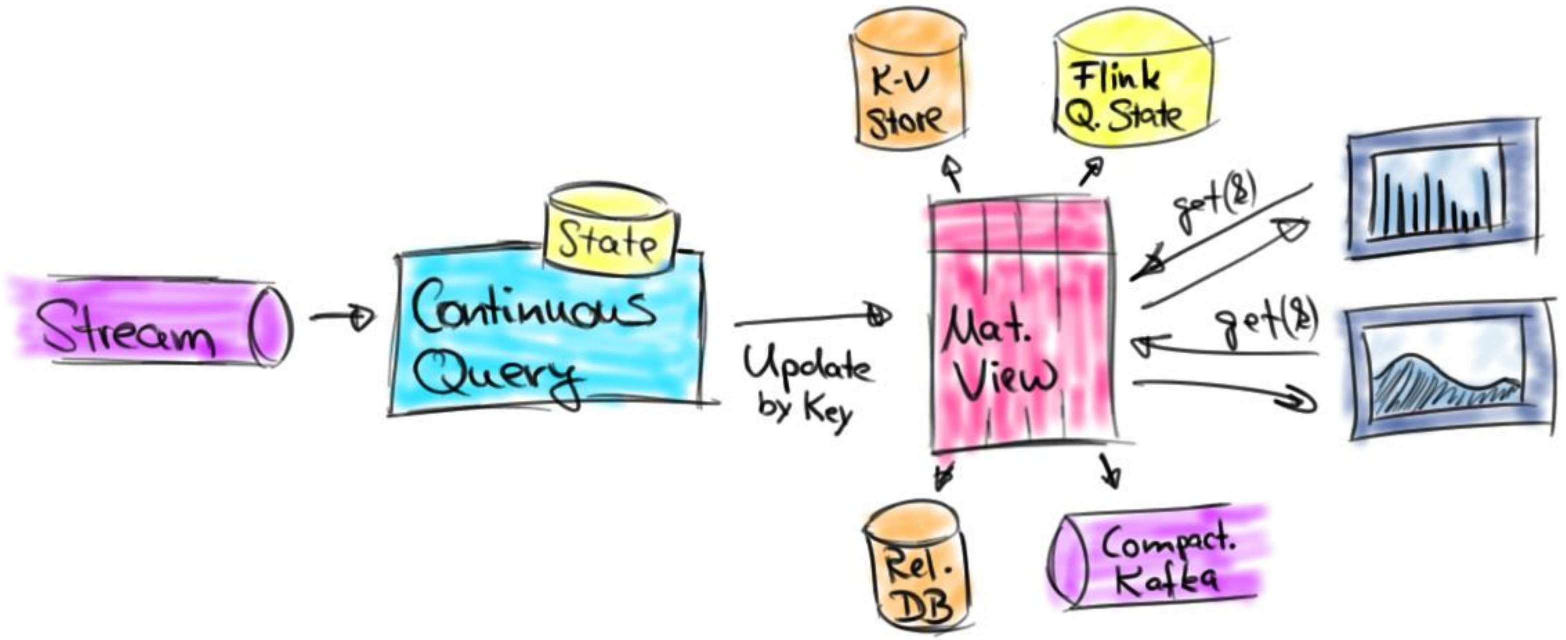
Data Pipelines

- 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

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



Training Environment

https://github.com/ververica/sql-training/

What You Will Learn in This Training?

- Querying streaming data with SQL
- Expressing common stream processing operations with SQL
  - Window aggregations, stream joins, and pattern matching
- Piping the results of continuous queries into Kafka
- Materializing the results of continuous queries in MySQL
- Using Flink's SQL CLI client

Training Scenario: Taxi Ride Data

- We are working with data about taxi rides in New York

- Three tables

`Rides`	One start and one end event for each ride
`Fares`	One payment event for each ride
`DriverChanges`	One event for each driver change of a taxi

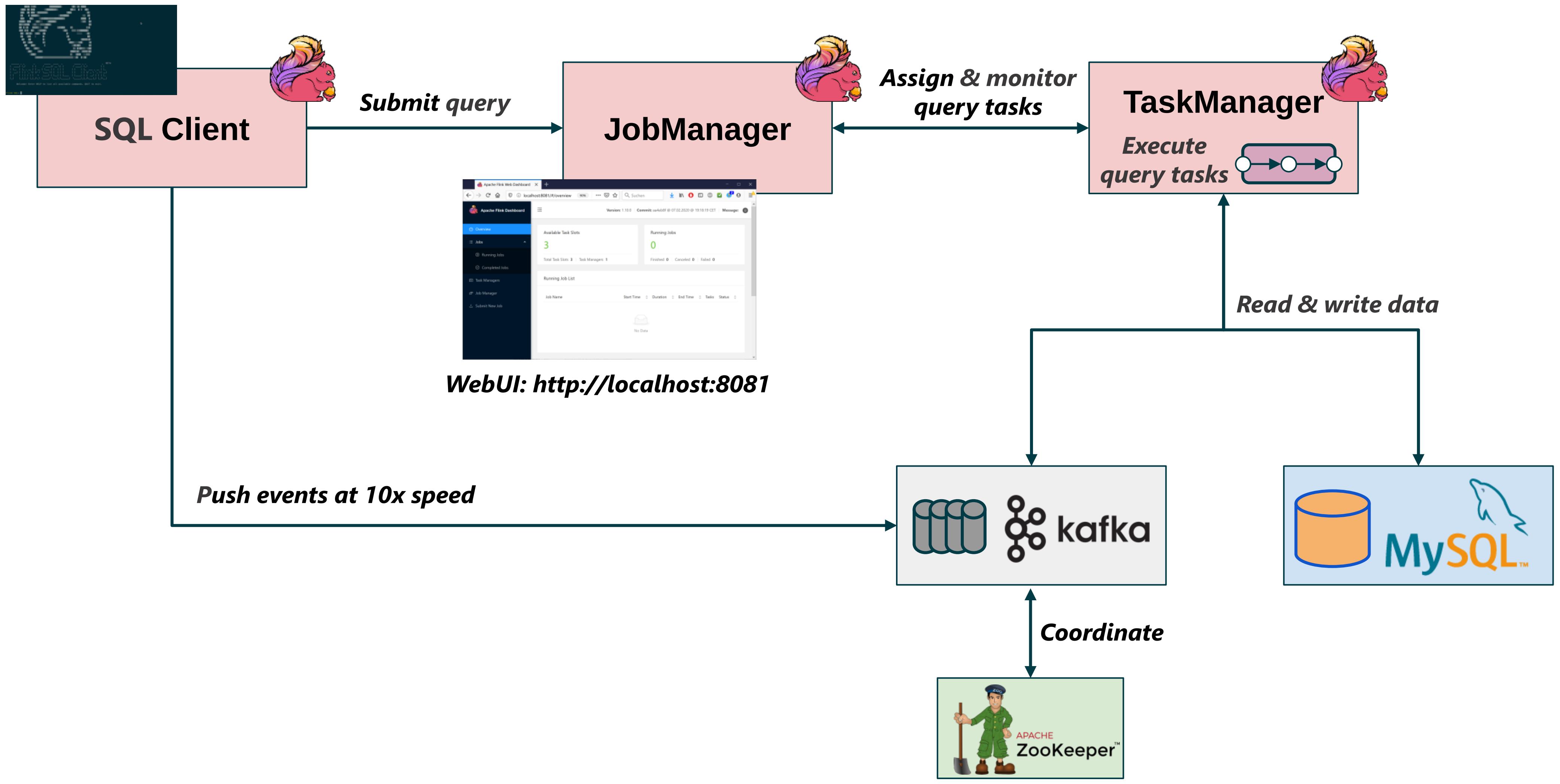
- All tables are registered and available in the environment

- Each tables is backed by a Kafka topic

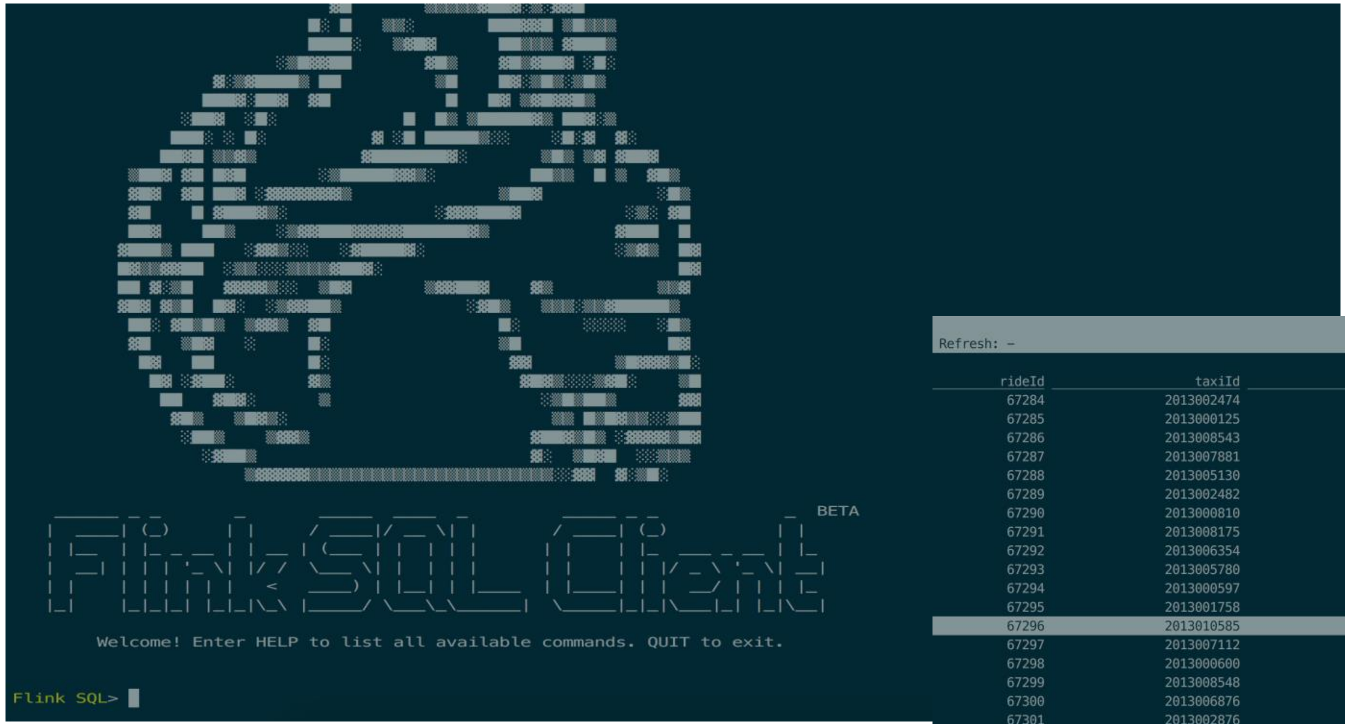
Training Scenario: Taxi Ride Data

```
Flink SQL> SELECT * FROM Rides;

rideId        taxiId     isStart          lon          lat          rideTime     psgCnt
     1    2013000001        true    -73.99078     40.76088  2013-01-01T00:00          1
     2    2013000002        true   -73.978325     40.77809  2013-01-01T00:00          5
     3    2013000003        true    -73.98962     40.72999  2013-01-01T00:00          1


Flink SQL> SELECT * FROM Fares;

rideId                payTime      payMethod        tip       toll      fare
    65    2013-01-01T00:00:36            CSH        0.0        0.0       3.5
   137       2013-01-01T00:01            CSH        0.0        0.0       3.5
    77    2013-01-01T00:01:22            CSH        0.0        0.0       4.0


Flink SQL> SELECT * FROM DriverChanges;

    taxiId      driverId          usageStartTime
2013000061    2013000061     2013-01-01T00:00:02
2013000062    2013000062     2013-01-01T00:00:03
2013000063    2013000063     2013-01-01T00:00:08
```

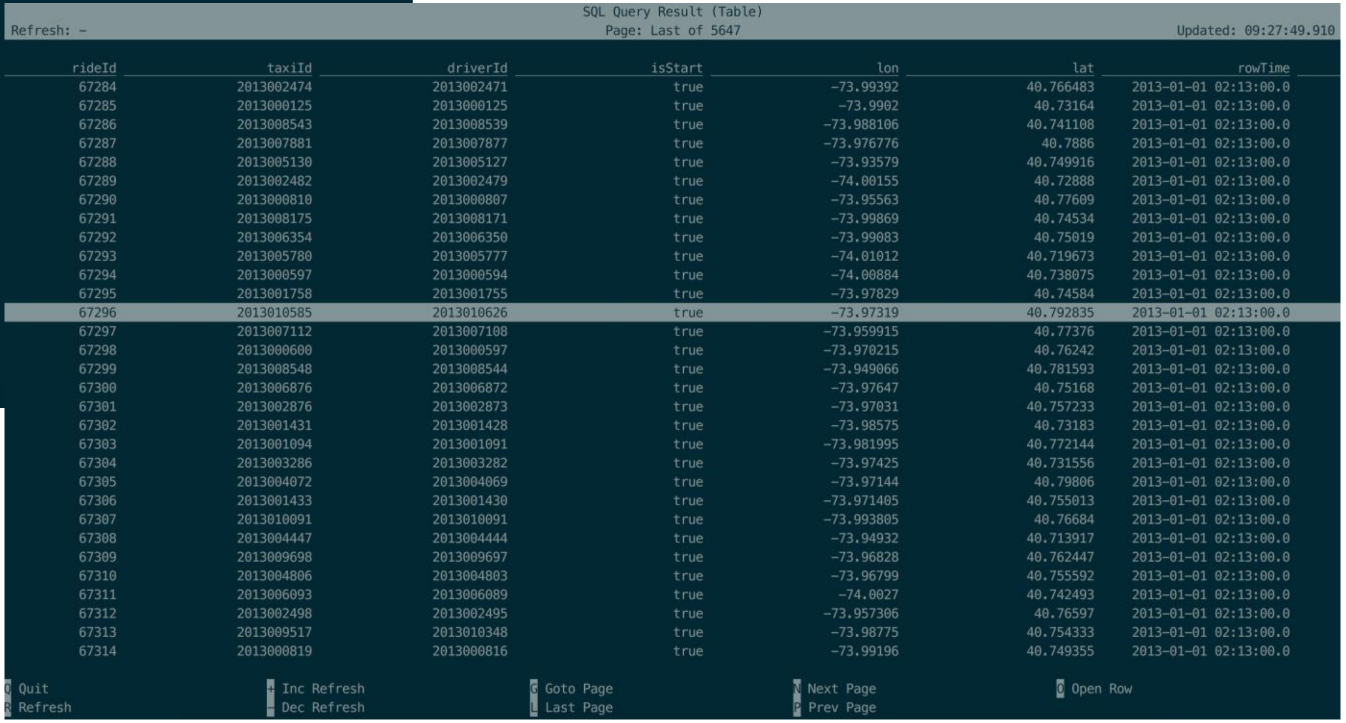
Our Training Environment



Introduction to SQL Client



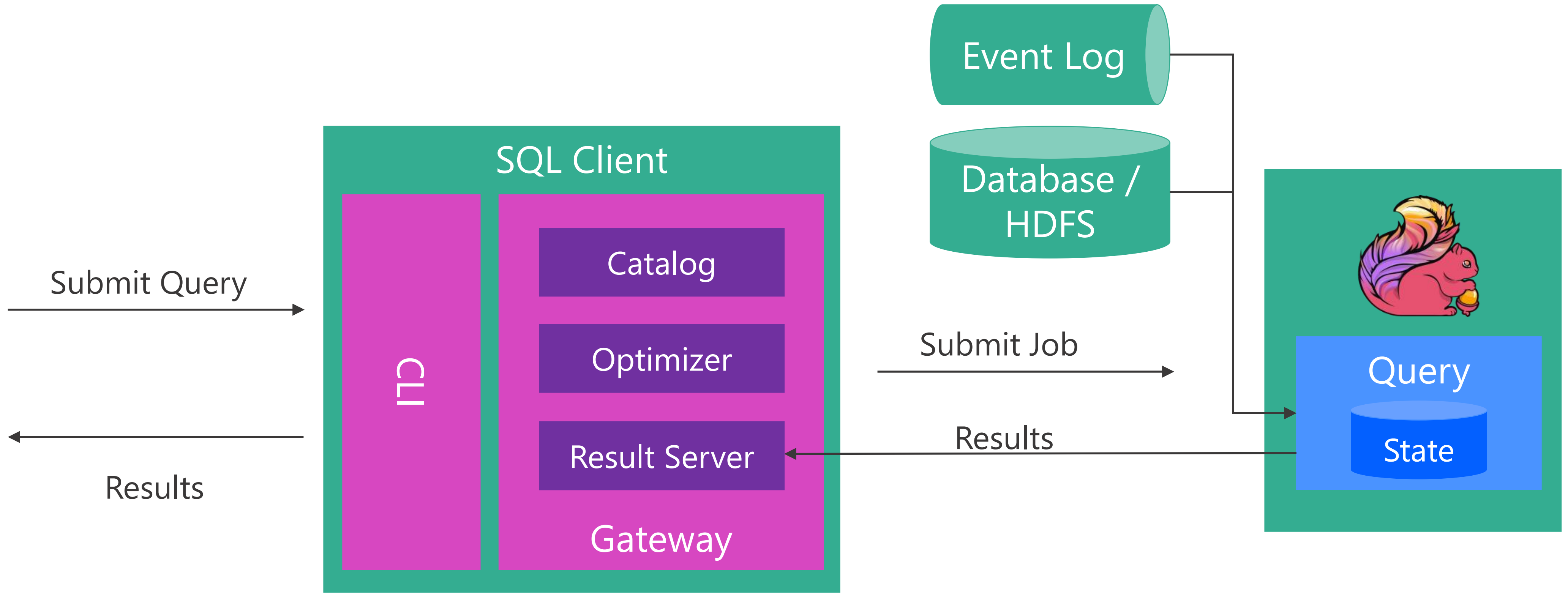
```
BETA

Welcome! Enter HELP to list all available commands. QUIT to exit.

Flink SQL> 
```



```
                                                  SQL Query Result (Table)
Refresh: -                                          Page: Last of 5647                                          Updated: 09:27:49.910

    rideId          taxiId        driverId        isStart             lon             lat                  rowTime
     67284      2013002474      2013002471           true       -73.99392       40.766483    2013-01-01 02:13:00.0
     67285      2013000125      2013000125           true        -73.9902        40.73164    2013-01-01 02:13:00.0
     67286      2013008543      2013008539           true      -73.988106       40.741108    2013-01-01 02:13:00.0
     67287      2013007881      2013007877           true      -73.976776         40.7886    2013-01-01 02:13:00.0
     67288      2013005130      2013005127           true       -73.93579       40.749916    2013-01-01 02:13:00.0
     67289      2013002482      2013002479           true       -74.00155        40.72888    2013-01-01 02:13:00.0
     67290      2013000810      2013000807           true       -73.95563        40.77609    2013-01-01 02:13:00.0
     67291      2013008175      2013008171           true       -73.99869        40.74534    2013-01-01 02:13:00.0
     67292      2013006354      2013006350           true       -73.99083        40.75019    2013-01-01 02:13:00.0
     67293      2013005780      2013005777           true       -74.01012       40.719673    2013-01-01 02:13:00.0
     67294      2013000597      2013000594           true       -74.00884       40.738075    2013-01-01 02:13:00.0
     67295      2013001758      2013001755           true       -73.97829        40.74584    2013-01-01 02:13:00.0
     67296      2013010585      2013010626           true       -73.97319       40.792835    2013-01-01 02:13:00.0
     67297      2013007112      2013007108           true      -73.959915        40.77376    2013-01-01 02:13:00.0
     67298      2013000600      2013000597           true      -73.970215        40.76242    2013-01-01 02:13:00.0
     67299      2013008548      2013008544           true      -73.949066       40.781593    2013-01-01 02:13:00.0
     67300      2013006876      2013006872           true       -73.97647        40.75168    2013-01-01 02:13:00.0
     67301      2013002876      2013002873           true       -73.97031       40.757233    2013-01-01 02:13:00.0
     67302      2013001431      2013001428           true       -73.98575        40.73183    2013-01-01 02:13:00.0
     67303      2013001094      2013001091           true      -73.981995       40.772144    2013-01-01 02:13:00.0
     67304      2013003286      2013003282           true       -73.97425       40.731556    2013-01-01 02:13:00.0
     67305      2013004072      2013004069           true       -73.97144        40.79806    2013-01-01 02:13:00.0
     67306      2013001433      2013001430           true      -73.971405       40.755013    2013-01-01 02:13:00.0
     67307      2013010091      2013010091           true      -73.993805        40.76684    2013-01-01 02:13:00.0
     67308      2013004447      2013004444           true       -73.94932       40.713917    2013-01-01 02:13:00.0
     67309      2013009698      2013009697           true       -73.96828       40.762447    2013-01-01 02:13:00.0
     67310      2013004806      2013004803           true       -73.96799       40.755592    2013-01-01 02:13:00.0
     67311      2013006093      2013006089           true        -74.0027       40.742493    2013-01-01 02:13:00.0
     67312      2013002498      2013002495           true      -73.957306        40.76597    2013-01-01 02:13:00.0
     67313      2013009517      2013010348           true       -73.98775       40.754333    2013-01-01 02:13:00.0
     67314      2013000819      2013000816           true       -73.99196       40.749355    2013-01-01 02:13:00.0

Q Quit              + Inc Refresh        G Goto Page        N Next Page        O Open Row
R Refresh           - Dec Refresh        L Last Page        P Prev Page
```

Interactive Query Submission via SQL Client

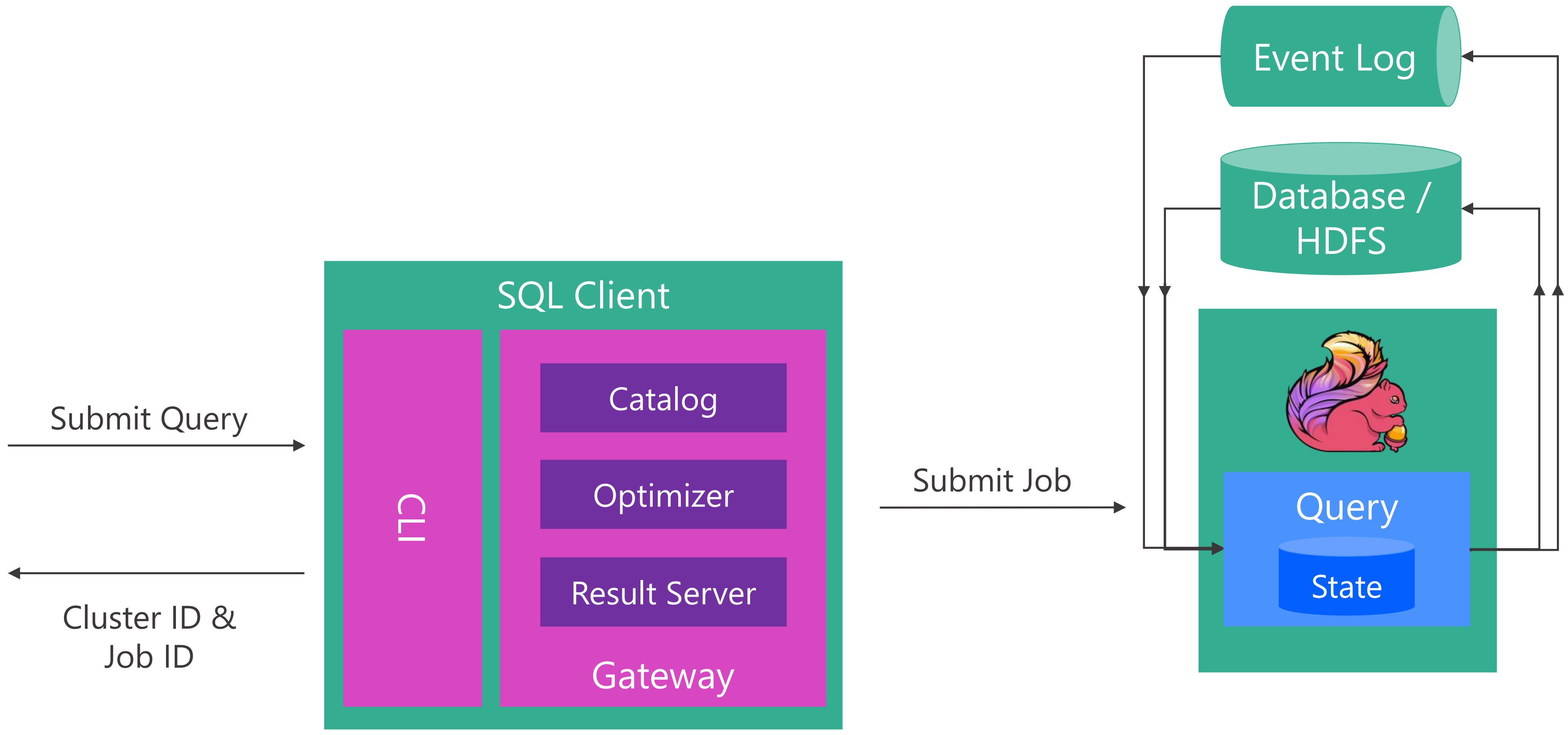
```
SELECT
  user,
  COUNT(url) AS cnt
FROM clicks
GROUP BY user
```



Changelog or Table

Detached Query Submission via SQL Client

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



Hands On Exercises

Introduction to SQL on Flink

Continue with the “Introduction to the Training Environment” in “Introduction to SQL on Flink”

https://github.com/ververica/sql-training/wiki/Introduction-to-SQL-on-Flink

We are here to help!

ververica

www.ververica.com

@VervericaData