timeplus | OSA CON 23

Query Live Data with SQL

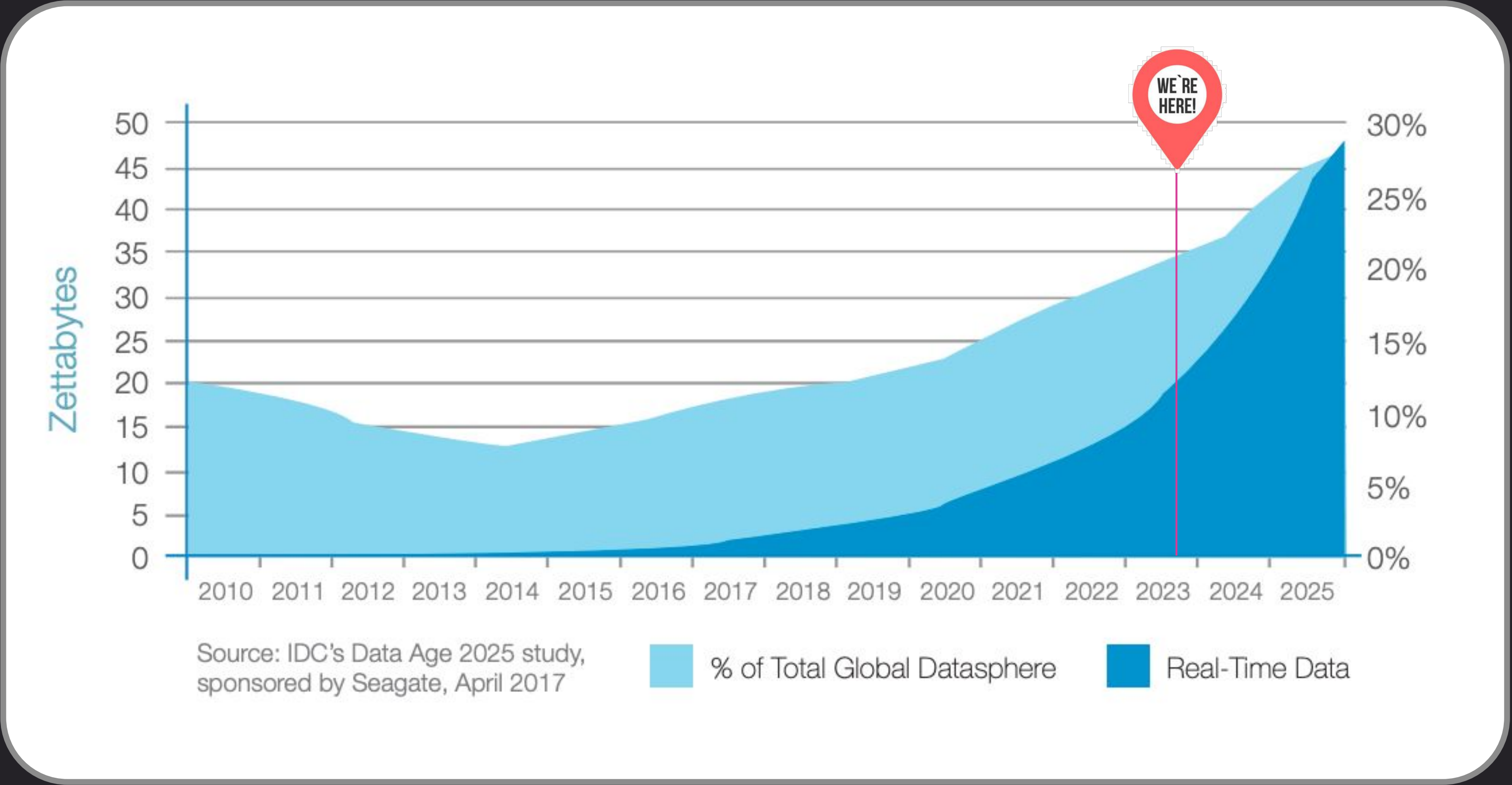
Why, how, and what's next?

Dec 12, 2023

Jove Zhong
Co-Founder and Head of Product, Timeplus

Gang Tao
Co-Founder and CTO, Timeplus

Live data is everywhere, at the edge and cloud



46 ZB

of data created by billions of IoT by 2025

30%

of data generated will be real-time by 2025

Only **1%**

of data is analyzed and streaming data is primarily untapped

kafka PULSAR Redpanda WarpStream Amazon Kinesis NATS MQTT

Live

Why SQL For ~~Big~~ Data?

Reliable

Fast

Easy

Powerful

Descriptive

Sample Use Cases

FinTech

- Real-time post-trade analytics
- Real-time pricing

DevOps

- Real-time Github insights
- Real-time o11y and usage based pricing

Security Compliance

- SOC2 compliance
- Container vulnerability monitoring
- Monitor Superblocks user activities
- Protect sensitive info in Slack

IoT

- Real-time fleet monitoring
- Oil well edge monitoring

Customer 360

- Auth0 notifications for new signups
- HubSpot custom dashboards/alerts
- Jitsu clickstream analytics
- Real-time Twitter marketing

Misc

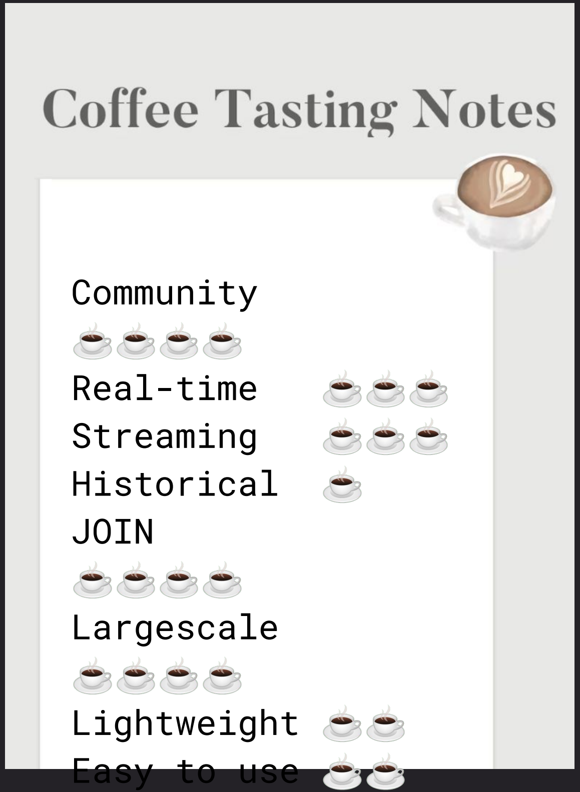
- Wildfire monitoring and alerting
- Data-driven parent

Learn more: https://docs.timeplus.com/showcases

How do you like your coffee?

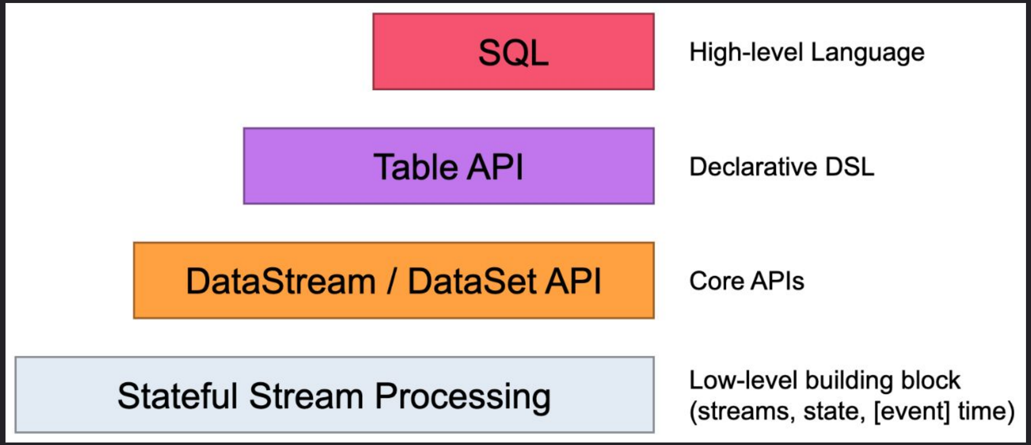
	Streaming Processor	Streaming Database	Real-Time Database
	Flink	ksqlDB, Hazelcast	Druid, Pinot, Trino
		proton	ClickHouse, StarRocks
	Arroyo	RisingWave	Databend

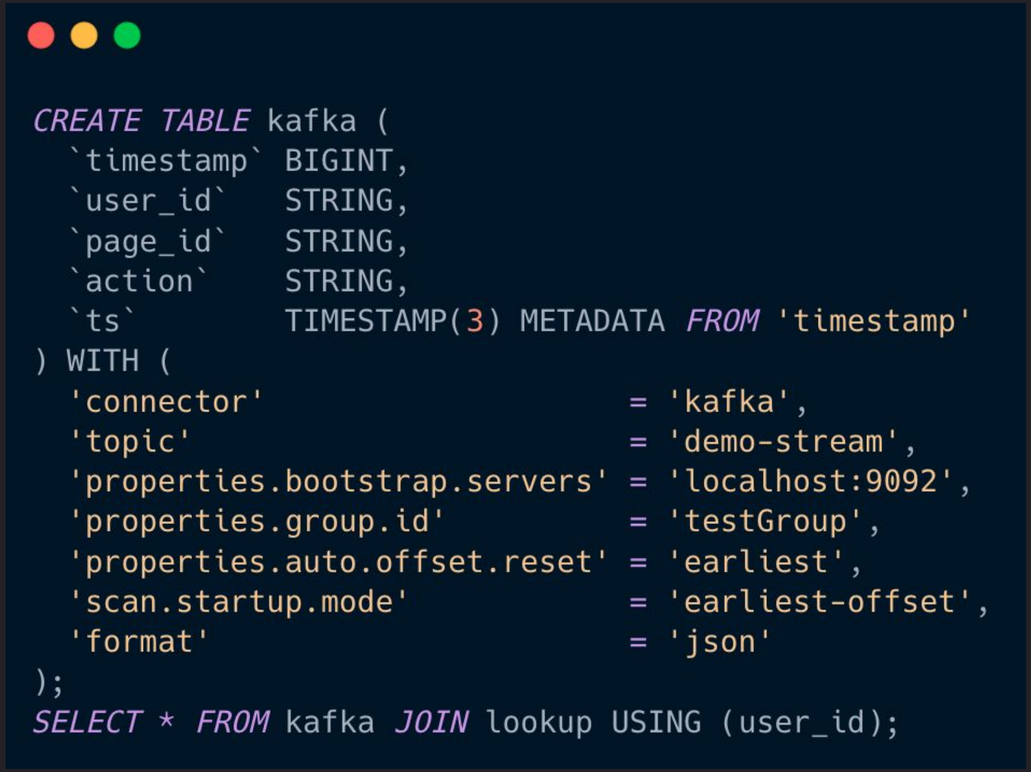
FlinkSQL
since 2016
espresso



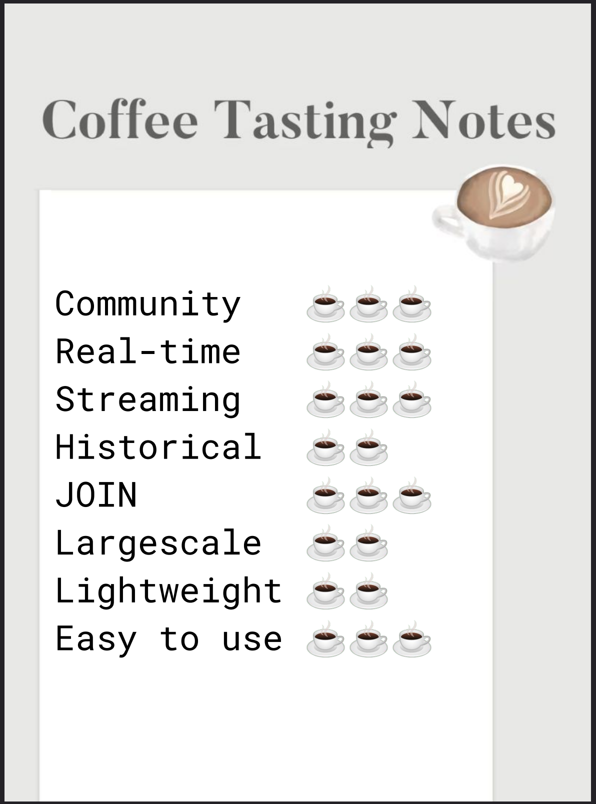
Coffee Tasting Notes

Community	☕☕☕☕
Real-time	☕☕☕
Streaming	☕☕☕
Historical	☕
JOIN	☕☕☕☕
Largescale	☕☕☕☕
Lightweight	☕☕
Easy to use	☕☕

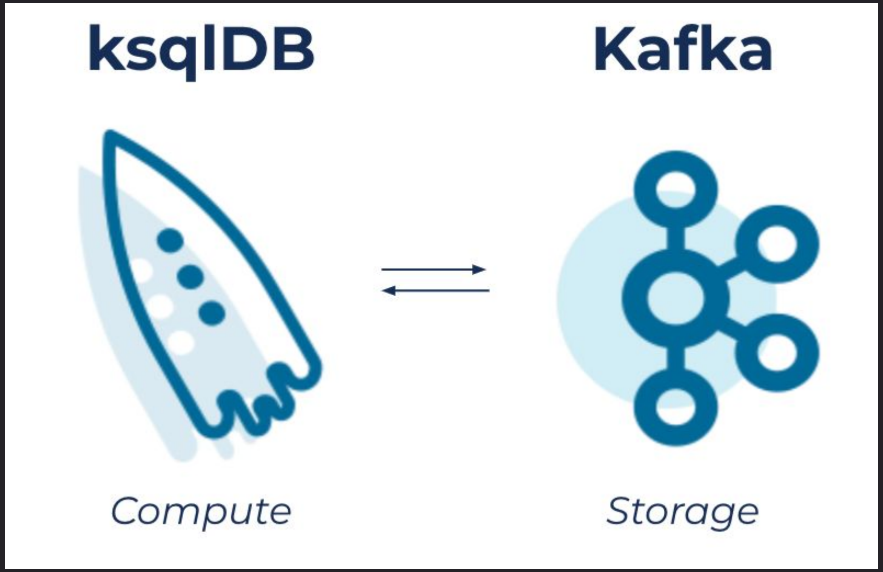

SQL
High-level Language
Table API
Declarative DSL
DataStream / DataSet API
Core APIs
Stateful Stream Processing
Low-level building block (streams, state, [event] time)

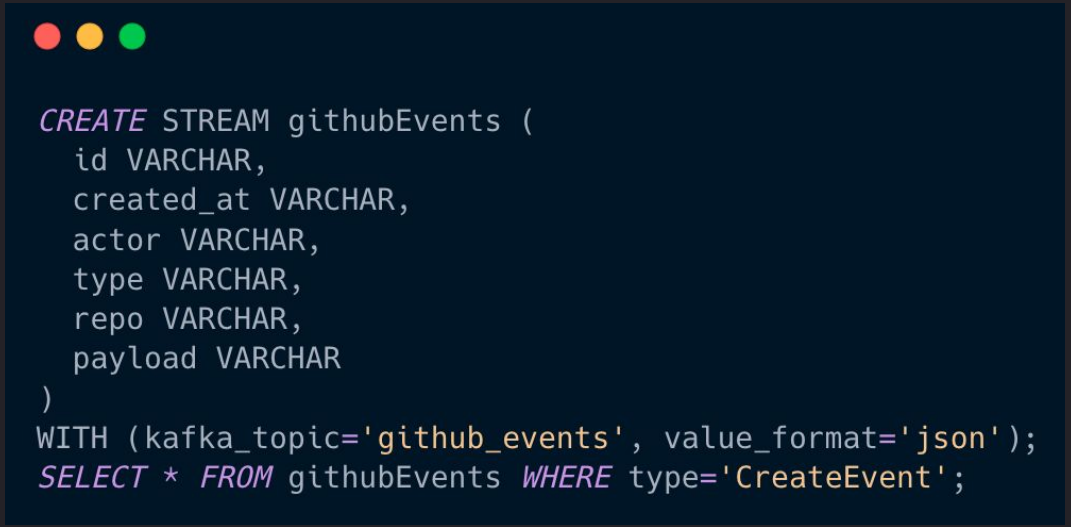


```sql
CREATE TABLE kafka (
  `timestamp` BIGINT,
  `user_id`   STRING,
  `page_id`   STRING,
  `action`    STRING,
  `ts`        TIMESTAMP(3) METADATA FROM 'timestamp'
) WITH (
  'connector'                    = 'kafka',
  'topic'                        = 'demo-stream',
  'properties.bootstrap.servers' = 'localhost:9092',
  'properties.group.id'          = 'testGroup',
  'properties.auto.offset.reset' = 'earliest',
  'scan.startup.mode'            = 'earliest-offset',
  'format'                       = 'json'
);
SELECT * FROM kafka JOIN lookup USING (user_id);
```


Coffee Tasting Notes

Community	☕☕☕
Real-time	☕☕☕
Streaming	☕☕☕
Historical	☕☕
JOIN	☕☕☕
Largescale	☕☕
Lightweight	☕☕
Easy to use	☕☕☕

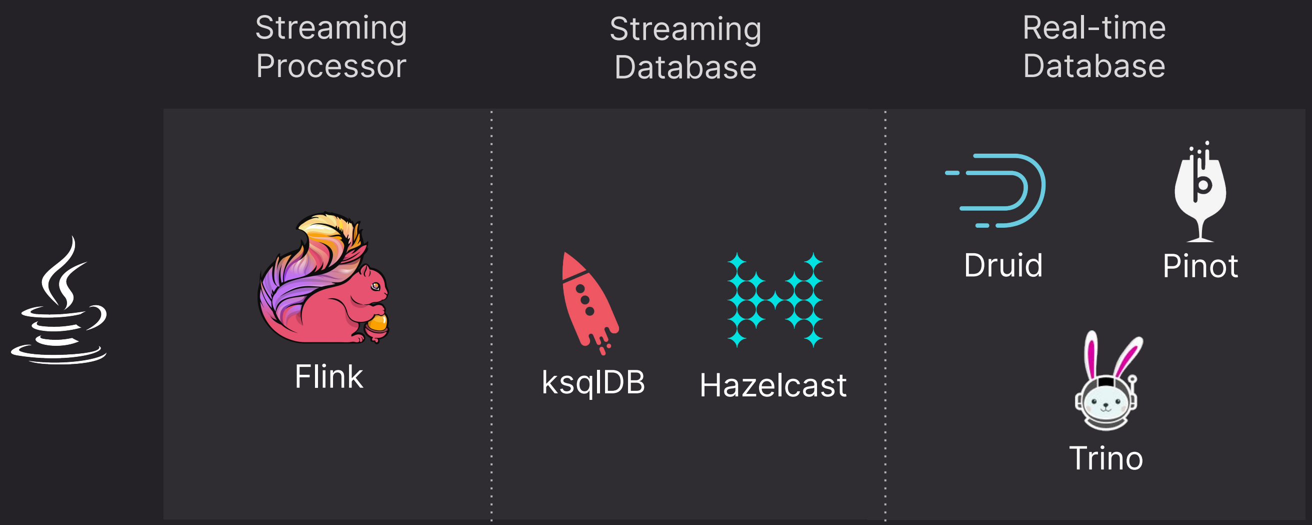


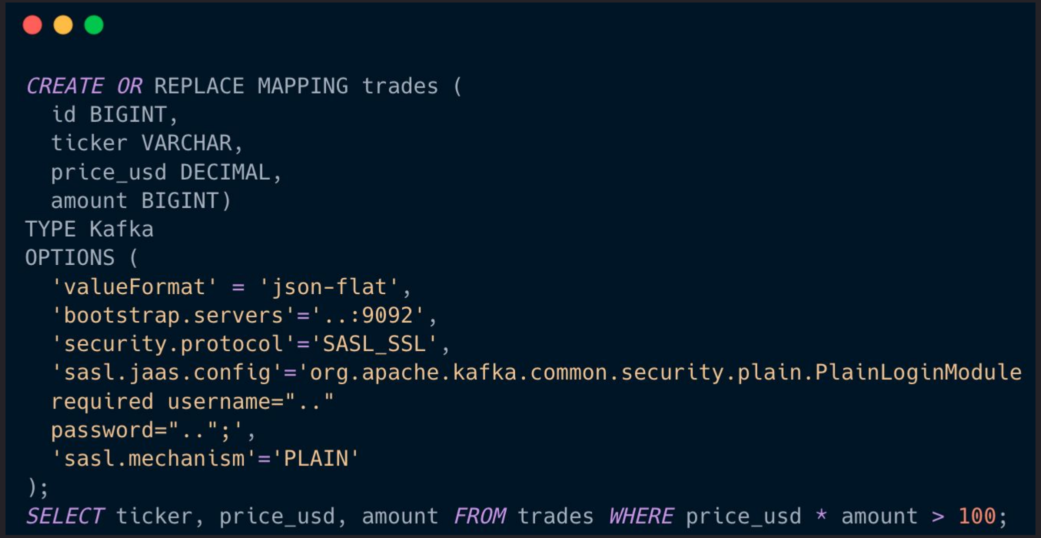

```
CREATE STREAM githubEvents (
  id VARCHAR,
  created_at VARCHAR,
  actor VARCHAR,
  type VARCHAR,
  repo VARCHAR,
  payload VARCHAR
)
WITH (kafka_topic='github_events', value_format='json');
SELECT * FROM githubEvents WHERE type='CreateEvent';
```

HAZELCAST

Distributed computation and storage platform

No dependency on disk storage, it keeps all its operational state in the RAM of the cluster.

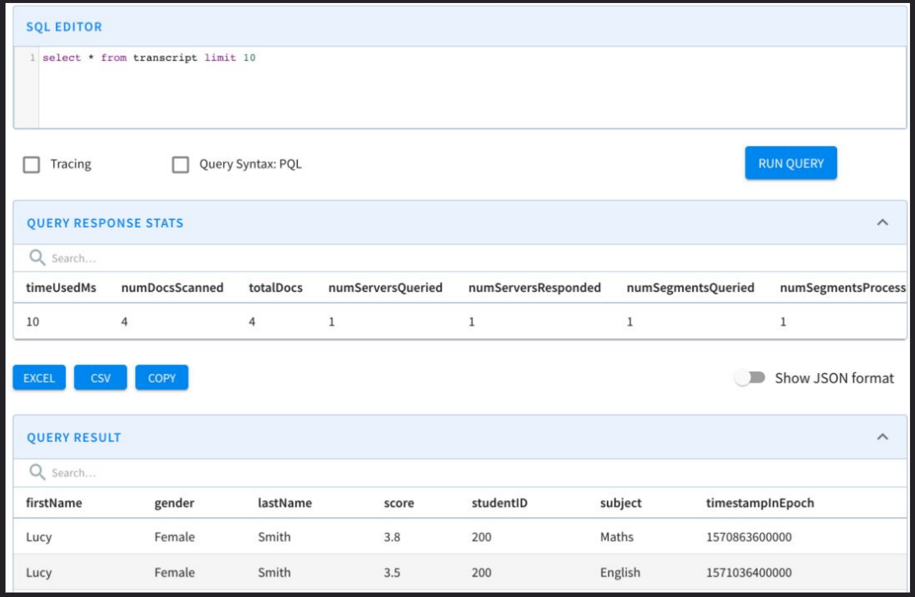




```
CREATE OR REPLACE MAPPING trades (
  id BIGINT,
  ticker VARCHAR,
  price_usd DECIMAL,
  amount BIGINT)
TYPE Kafka
OPTIONS (
  'valueFormat' = 'json-flat',
  'bootstrap.servers'='..:9092',
  'security.protocol'='SASL_SSL',
  'sasl.jaas.config'='org.apache.kafka.common.security.plain.PlainLoginModule
  required username=".."
  password="..";',
  'sasl.mechanism'='PLAIN'
);
SELECT ticker, price_usd, amount FROM trades WHERE price_usd * amount > 100;
```

pinot

1. create a schema json (columns, PKs)
2. create a table configuration json (streamType=Kafka)
3. docker run .. apachepinot/pinot:latest AddTable \
   -schemaFile /tmp/transcript-schema.json \
   -tableConfigFile /tmp/transcript-table-realtime.json \
   ..
   -exec



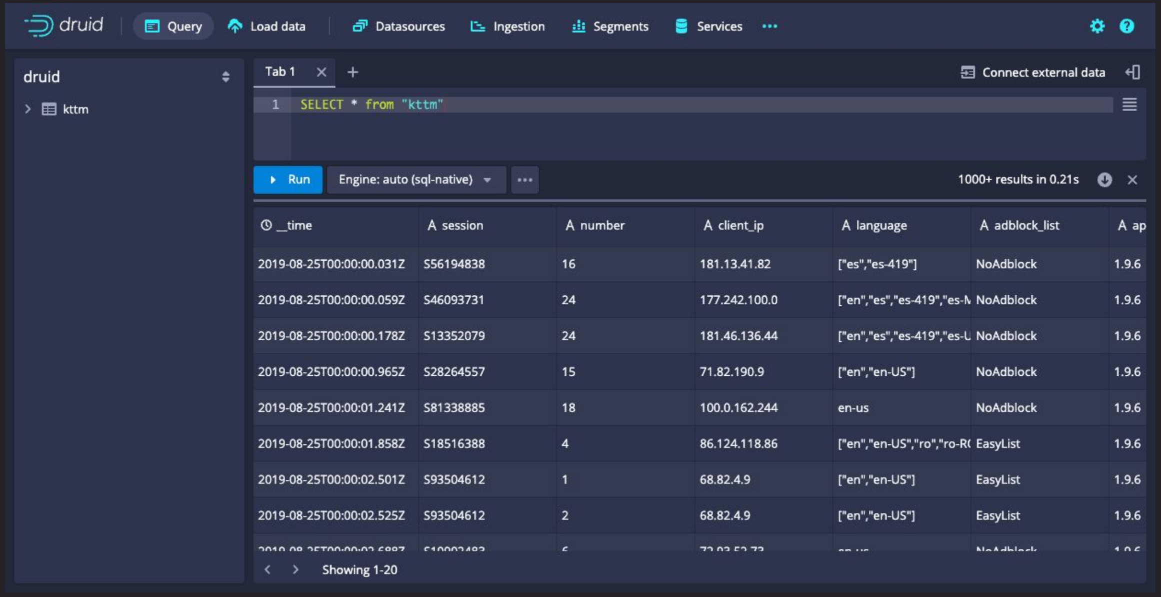
```
select * from transcript limit 10
```

timeUsedMs	numDocsScanned	totalDocs	numServersQueried	numServersResponded	numSegmentsQueried	numSegmentsProcess
10	4	4	1	1	1	1

firstName	gender	lastName	score	studentID	subject	timestampInEpoch
Lucy	Female	Smith	3.8	200	Maths	1570863600000
Lucy	Female	Smith	3.5	200	English	1571036400000

druid

1. load the druid-kafka-indexing-service extension on both the Overlord and the MiddleManagers
2. Create a supervisor-spec.json containing the Kafka supervisor spec file.
3. curl -X POST -H 'Content-Type: application/json' -d @supervisor-spec.json http://localhost:8090/druid/indexer/v1/supervisor

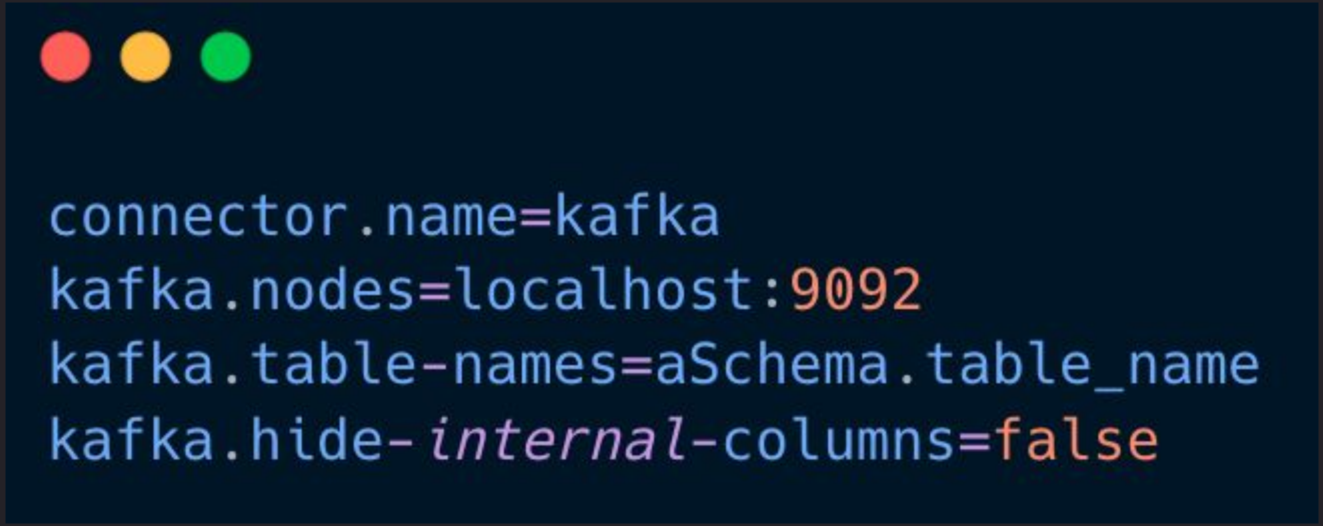


```
SELECT * from "kttm"
```

1000+ results in 0.21s

_time	session	number	client_ip	language	adblock_list	ap
2019-08-25T00:00:00.031Z	S56194838	16	181.13.41.82	["es","es-419"]	NoAdblock	1.9.6
2019-08-25T00:00:00.059Z	S46093731	24	177.242.100.0	["en","es","es-419","es-M	NoAdblock	1.9.6
2019-08-25T00:00:00.178Z	S13352079	24	181.46.136.44	["en","es","es-419","es-U	NoAdblock	1.9.6
2019-08-25T00:00:00.965Z	S28264557	15	71.82.190.9	["en","en-US"]	NoAdblock	1.9.6
2019-08-25T00:00:01.241Z	S81338885	18	100.0.162.244	en-us	NoAdblock	1.9.6
2019-08-25T00:00:01.858Z	S18516388	4	86.124.118.86	["en","en-US","ro","ro-R(	EasyList	1.9.6
2019-08-25T00:00:02.501Z	S93504612	1	68.82.4.9	["en","en-US"]	EasyList	1.9.6
2019-08-25T00:00:02.525Z	S93504612	2	68.82.4.9	["en","en-US"]	EasyList	1.9.6

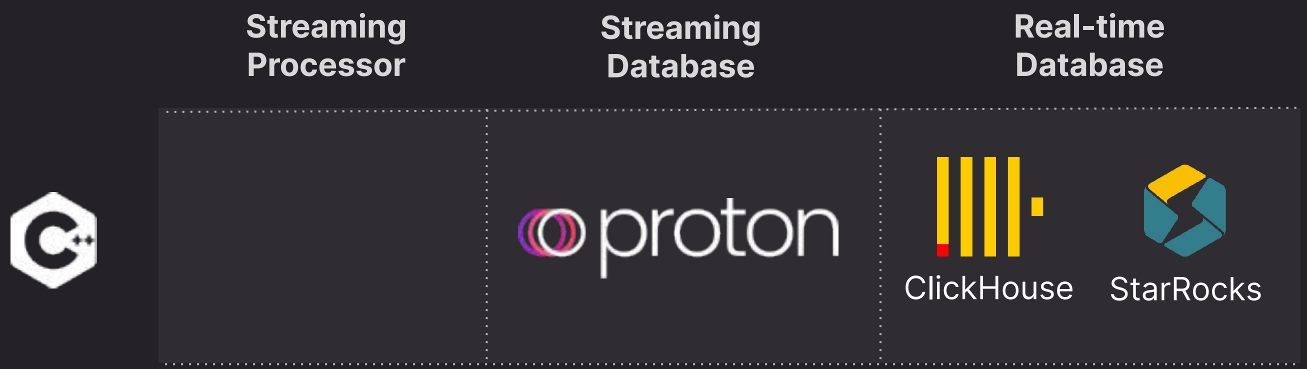
Add a catalog properties file etc/catalog/kafka.properties
for the Kafka connector.



```
connector.name=kafka
kafka.nodes=localhost:9092
kafka.table-names=aSchema.table_name
kafka.hide-internal-columns=false
```

```
$ ./trino --catalog kafka --schema aSchema

trino:aSchema> SELECT count(*) FROM customer;
```

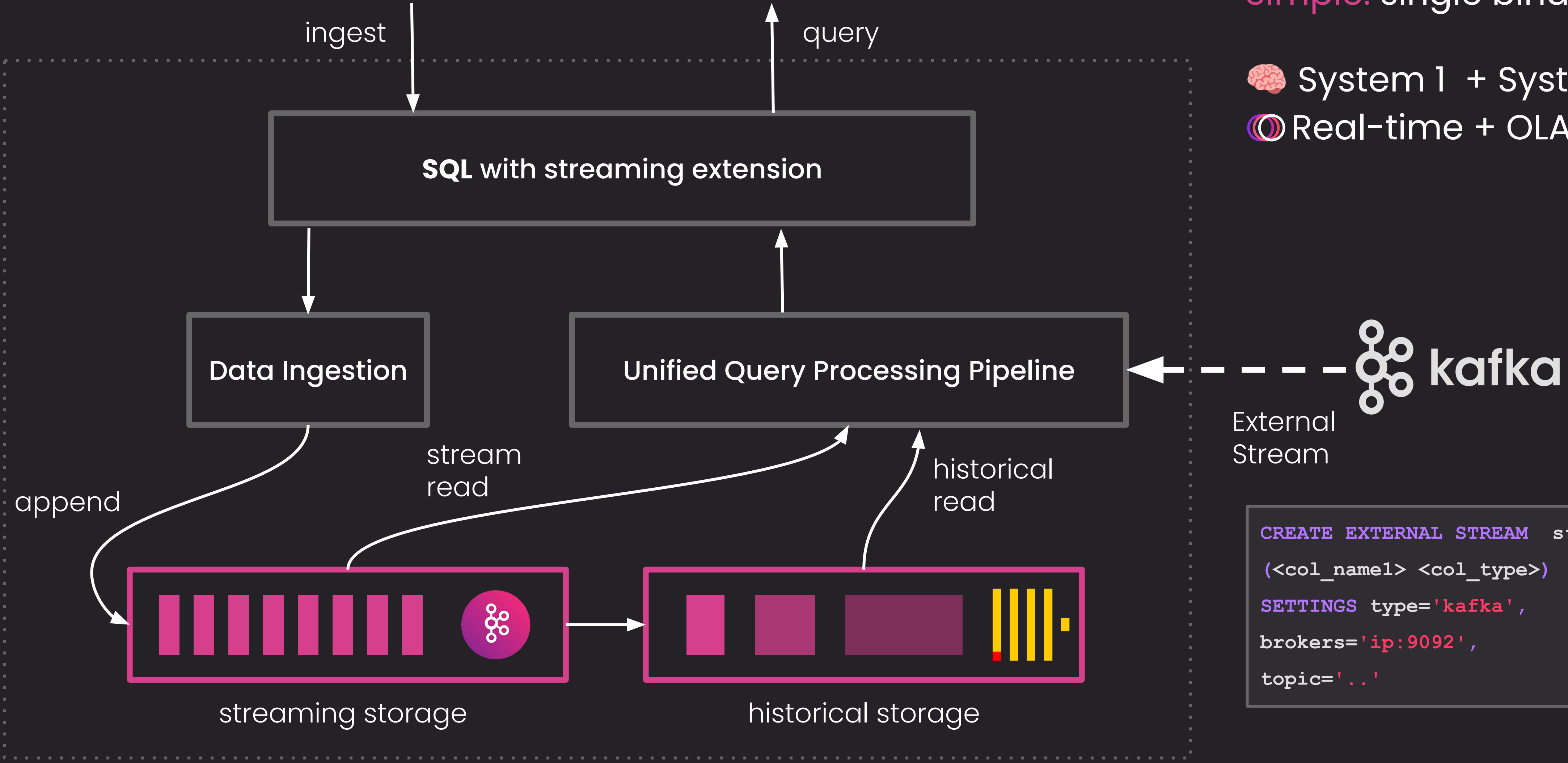

Streaming Processor
Streaming Database
Real-time Database
proton
ClickHouse
StarRocks

proton

the Next-Generation Streaming Database
(Kafka + Flink + ClickHouse)

Open: Apache v2 license
Fast: 4ms latency, 10m eps
Simple: single binary, SQL

🧠 System 1 + System 2
Real-time + OLAP



```
CREATE EXTERNAL STREAM  stream_name
(<col_name1> <col_type>)
SETTINGS type='kafka',
brokers='ip:9092',
topic='..'
```

proton

Stream tail

```
SELECT * FROM car_live_data
```

Historical query

```
SELECT * FROM table(car_live_data)
```

Global aggregation

```
SELECT count(*) FROM car_live_data
```

Window aggregation

```
SELECT window_start, count(*)
FROM tumble(car_live_data, 1m)
GROUP BY window_start
```

Sub streams

```
SELECT cid,
       speed_kmh,
       lag(speed_kmh) OVER
       (PARTITION BY cid) AS last_spd
FROM car_live_data
```

Late event

```
SELECT window_start, count(*)
FROM tumble(car_live_data, 5s)
GROUP BY window_start
EMIT AFTER WATERMARK AND DELAY 2s
```

Time travel

```
SELECT *
FROM car_live_data
WHERE
 _tp_time > now() - 1d
```

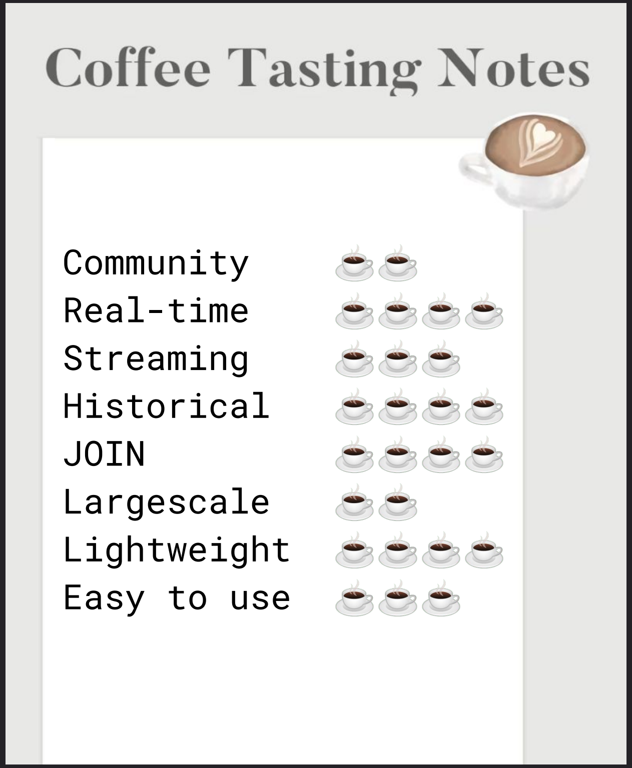
Stream join

```
SELECT
 device, cpu_usage, timestamp
FROM
 device_utils
INNER JOIN
table(device_products_info) AS dim
ON device_utils.product_id = dim.id
```

proton

Since 2021

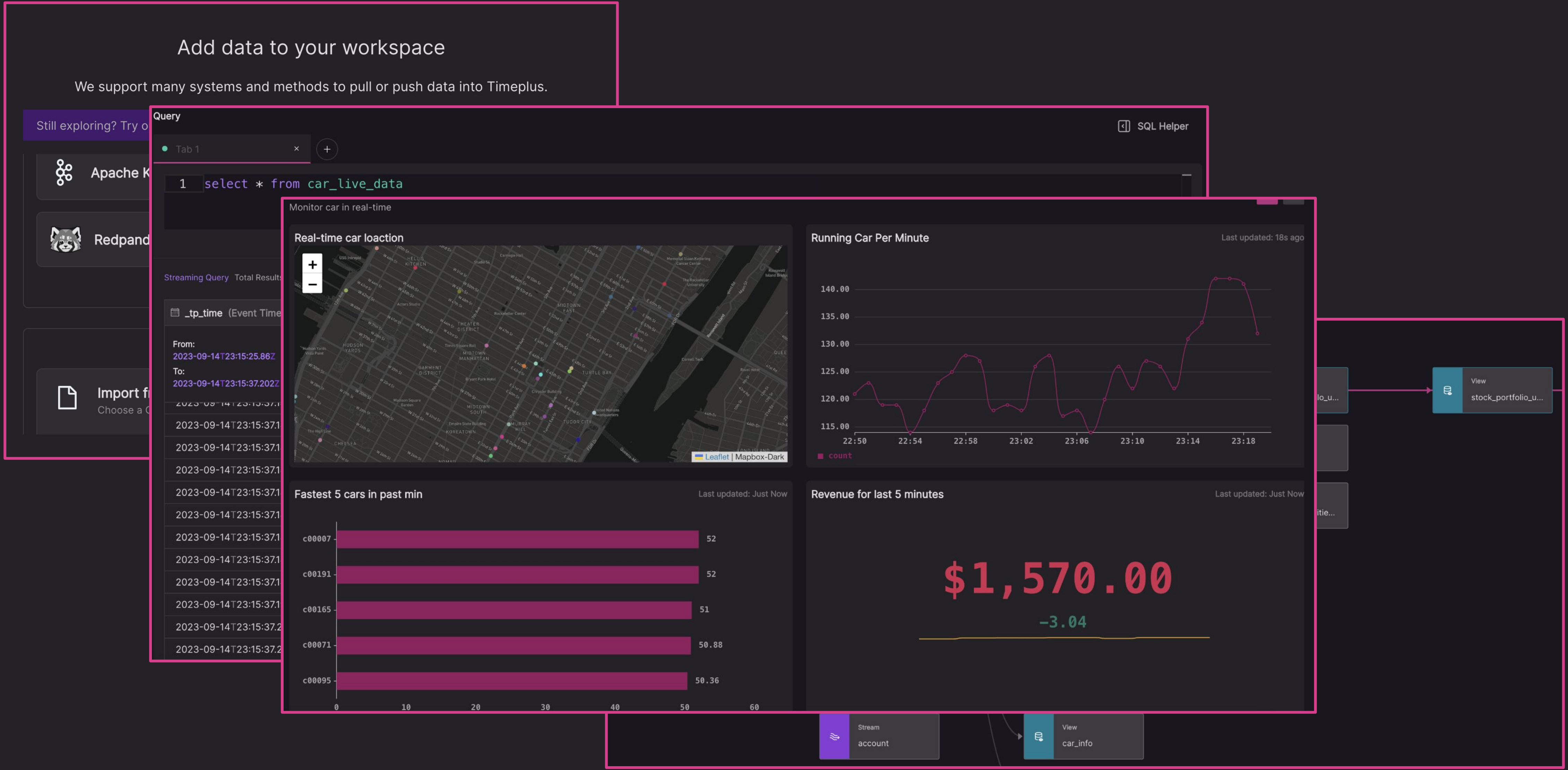
Mocha



Coffee Tasting Notes

Community	☕☕
Real-time	☕☕☕☕
Streaming	☕☕☕
Historical	☕☕☕☕
JOIN	☕☕☕☕
Largescale	☕☕
Lightweight	☕☕☕☕
Easy to use	☕☕☕

proton → timeplus



Add data to your workspace

We support many systems and methods to pull or push data into Timeplus.

```
select * from car_live_data
```

From: 2023-09-14T23:15:25.86Z
To: 2023-09-14T23:15:37.202Z

```
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.1
2023-09-14T23:15:37.2
2023-09-14T23:15:37.2
```

Monitor car in real-time

Real-time car loaction

Running Car Per Minute — Last updated: 18s ago

Fastest 5 cars in past min — Last updated: Just Now

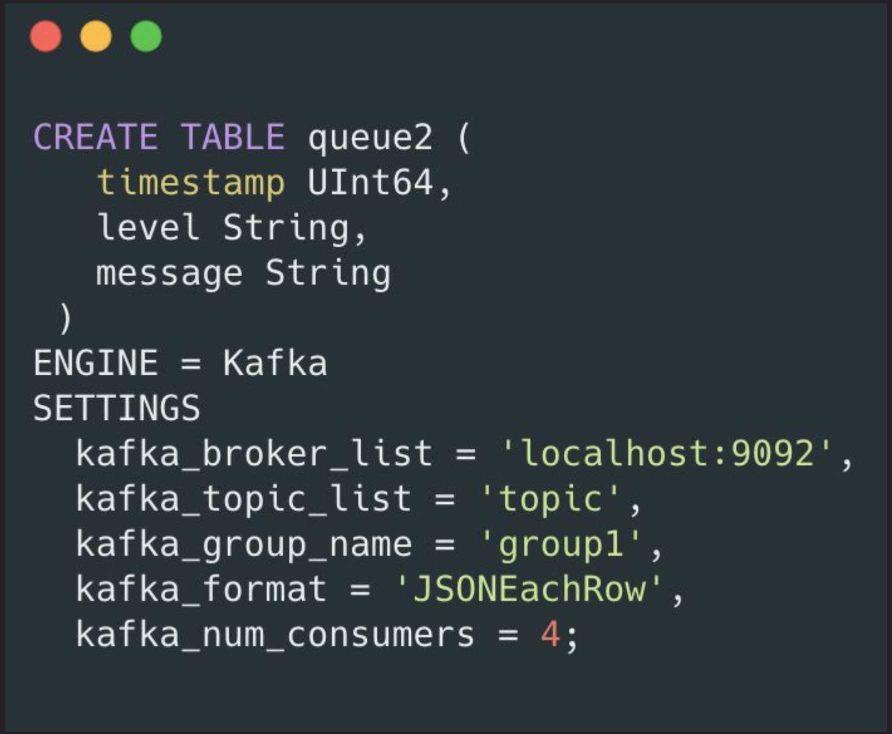
c00007	52
c00191	52
c00165	51
c00071	50.88
c00095	50.36

Revenue for last 5 minutes — Last updated: Just Now

$1,570.00

-3.04

ClickHouse

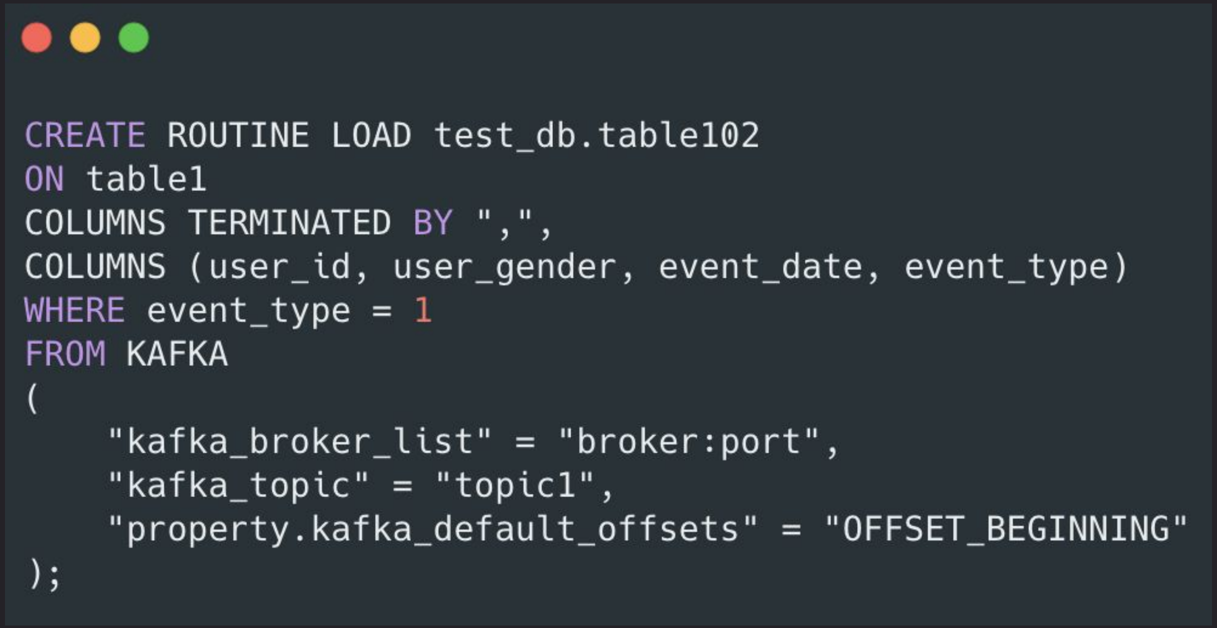


```sql
CREATE TABLE queue2 (
   timestamp UInt64,
   level String,
   message String
 )
ENGINE = Kafka
SETTINGS
  kafka_broker_list = 'localhost:9092',
  kafka_topic_list = 'topic',
  kafka_group_name = 'group1',
  kafka_format = 'JSONEachRow',
  kafka_num_consumers = 4;
```

ClickHouse features highlights

- Table engine and table function
- Rich functions 1500+
- Rich data types - Array, Map etc

StarRocks



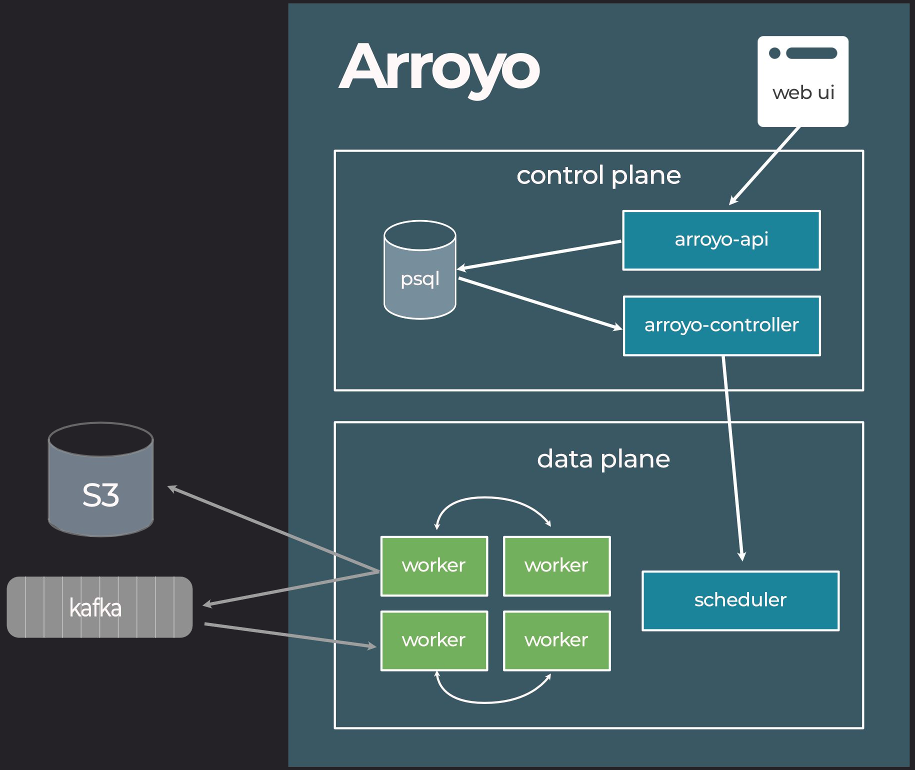
```sql
CREATE ROUTINE LOAD test_db.table102
ON table1
COLUMNS TERMINATED BY ",",
COLUMNS (user_id, user_gender, event_date, event_type)
WHERE event_type = 1
FROM KAFKA
(
    "kafka_broker_list" = "broker:port",
    "kafka_topic" = "topic1",
    "property.kafka_default_offsets" = "OFFSET_BEGINNING"
);
```

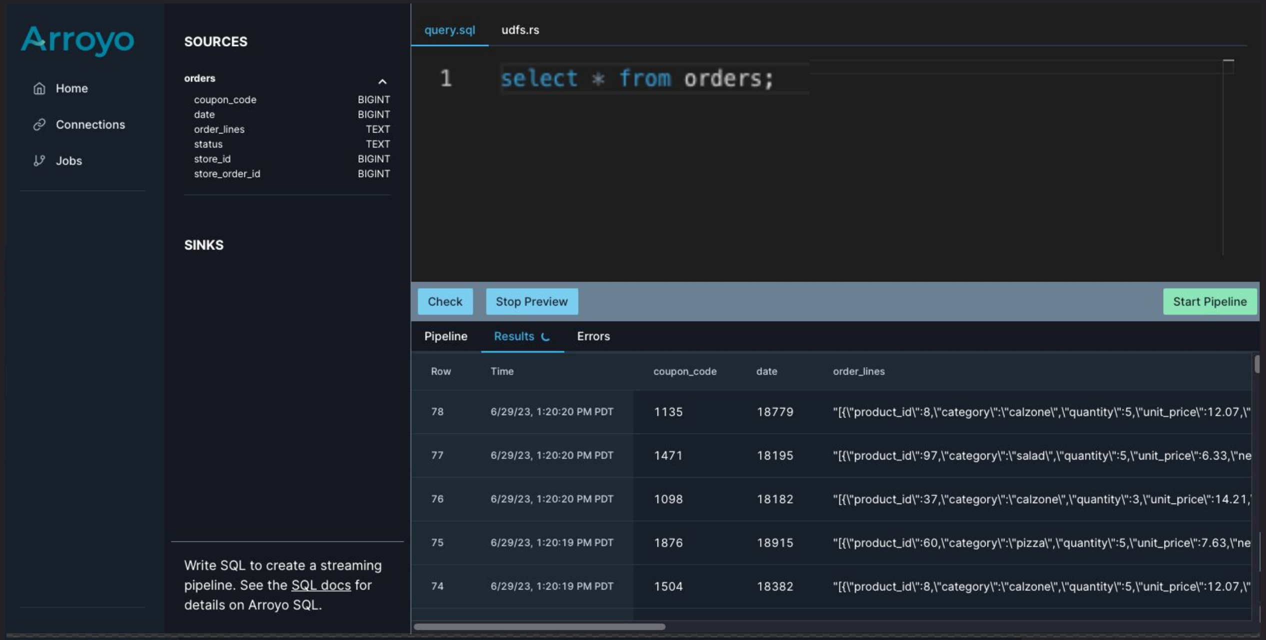
StarRocks features highlights

- More capable of joins
- High concurrency
- High frequency changes

Streaming Processor	Streaming Database	Real-time Database
Arroyo	RisingWave	Databend

Arroyo





SOURCES

orders

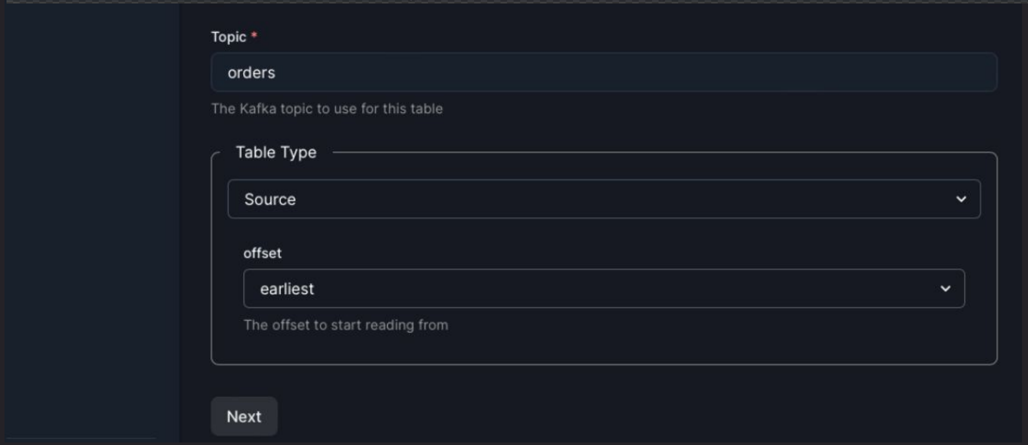
coupon_code	BIGINT
date	BIGINT
order_lines	TEXT
status	TEXT
store_id	BIGINT
store_order_id	BIGINT

SINKS

Write SQL to create a streaming pipeline. See the SQL docs for details on Arroyo SQL.

```sql
select * from orders;
```

Row	Time	coupon_code	date	order_lines
78	6/29/23, 1:20:20 PM PDT	1135	18779	"[{\"product_id\":8,\"category\":\"calzone\",\"quantity\":5,\"unit_price\":12.07,\"
77	6/29/23, 1:20:20 PM PDT	1471	18195	"[{\"product_id\":97,\"category\":\"salad\",\"quantity\":5,\"unit_price\":6.33,\"ne
76	6/29/23, 1:20:20 PM PDT	1098	18182	"[{\"product_id\":37,\"category\":\"calzone\",\"quantity\":3,\"unit_price\":14.21,
75	6/29/23, 1:20:19 PM PDT	1876	18915	"[{\"product_id\":60,\"category\":\"pizza\",\"quantity\":5,\"unit_price\":7.63,\"ne
74	6/29/23, 1:20:19 PM PDT	1504	18382	"[{\"product_id\":8,\"category\":\"calzone\",\"quantity\":5,\"unit_price\":12.07,\"



Topic *

orders

The Kafka topic to use for this table

Table Type

Source

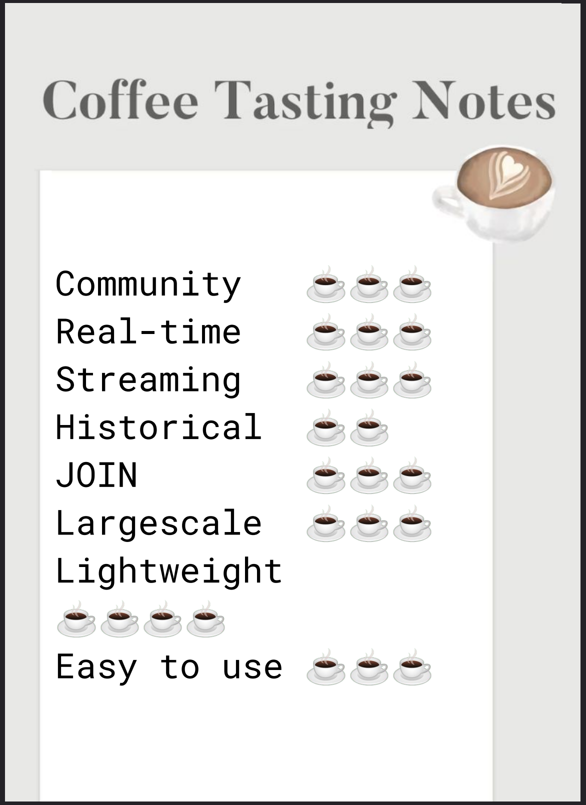
offset

earliest

The offset to start reading from

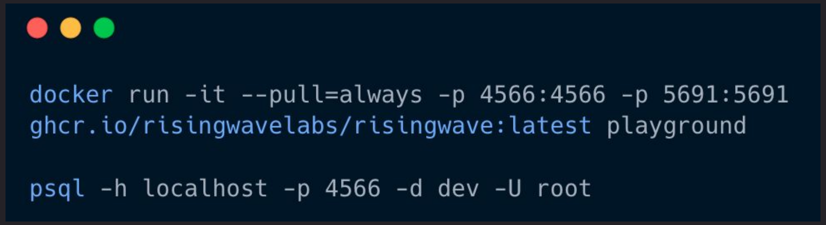
RisingWave

Cappuccino



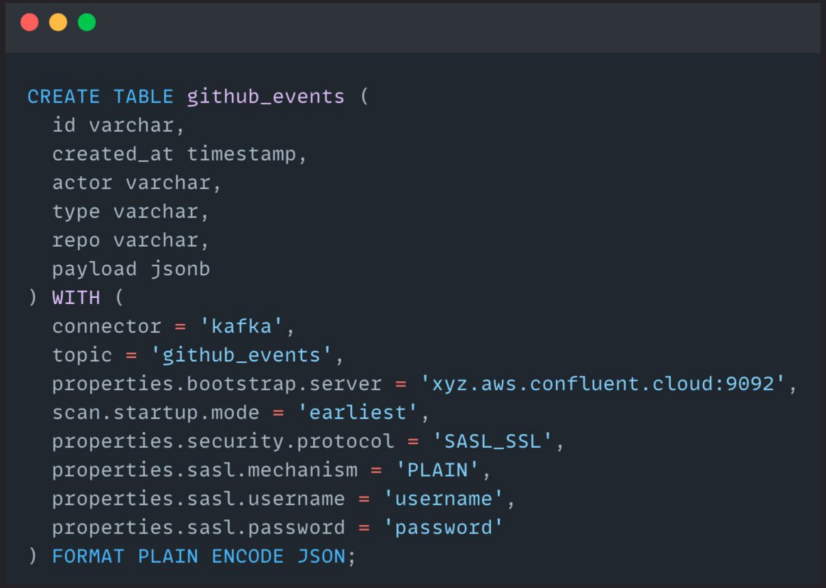
Coffee Tasting Notes

Community	☕☕☕
Real-time	☕☕☕
Streaming	☕☕☕
Historical	☕☕
JOIN	☕☕☕
Largescale	☕☕☕
Lightweight	☕☕☕☕
Easy to use	☕☕☕

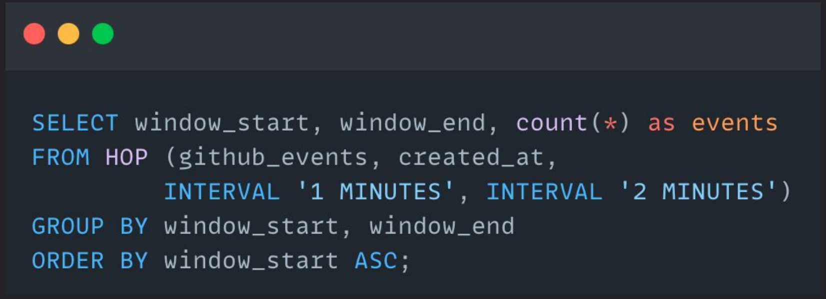


```
docker run -it --pull=always -p 4566:4566 -p 5691:5691
ghcr.io/risingwavelabs/risingwave:latest playground

psql -h localhost -p 4566 -d dev -U root
```

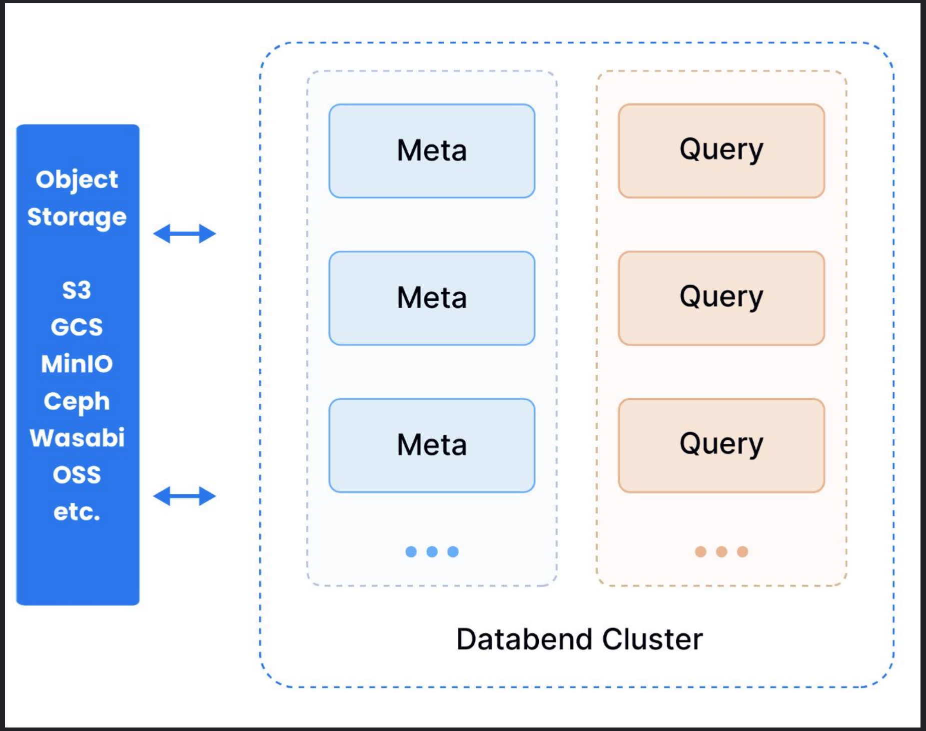


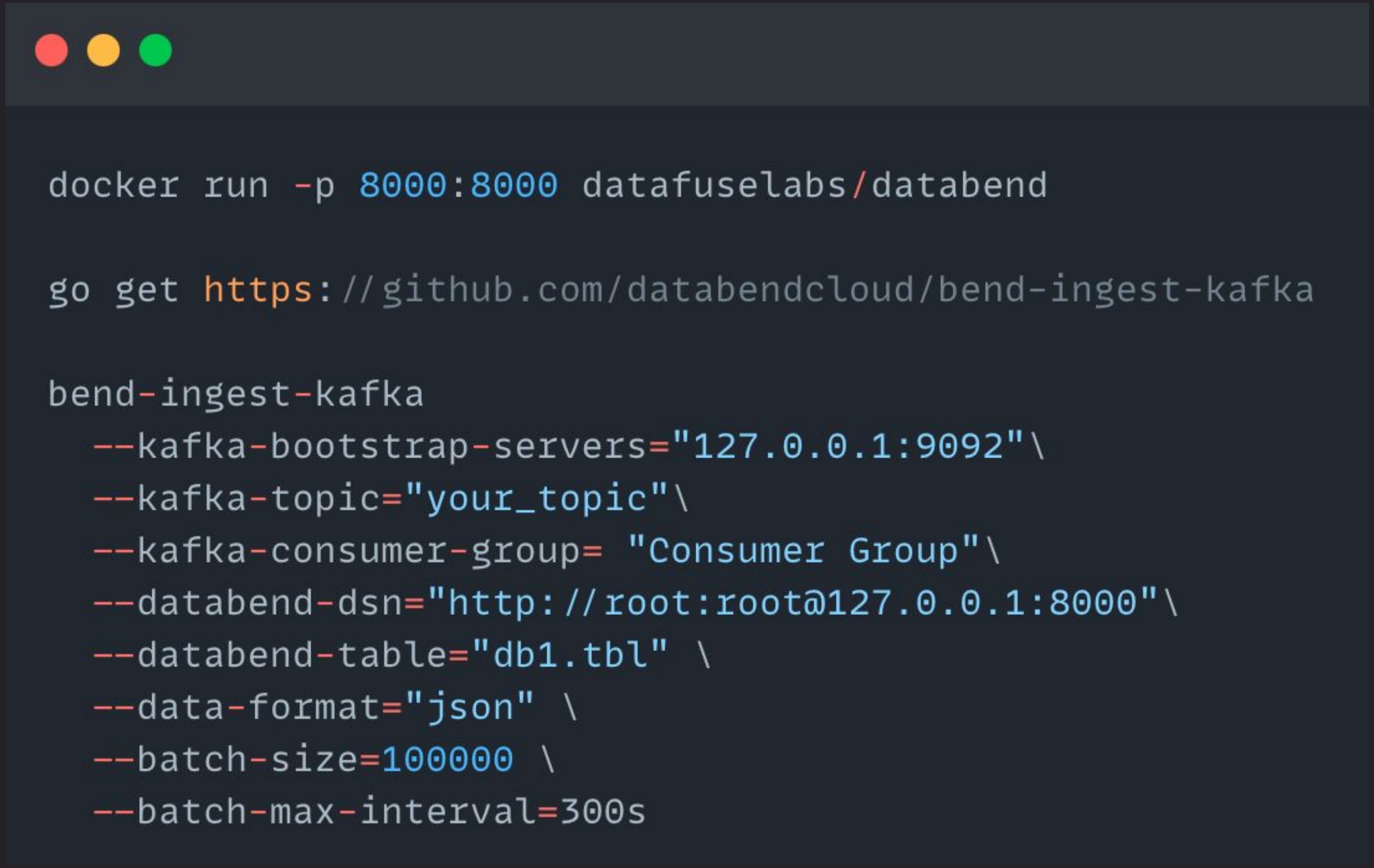
```
CREATE TABLE github_events (
  id varchar,
  created_at timestamp,
  actor varchar,
  type varchar,
  repo varchar,
  payload jsonb
) WITH (
  connector = 'kafka',
  topic = 'github_events',
  properties.bootstrap.server = 'xyz.aws.confluent.cloud:9092',
  scan.startup.mode = 'earliest',
  properties.security.protocol = 'SASL_SSL',
  properties.sasl.mechanism = 'PLAIN',
  properties.sasl.username = 'username',
  properties.sasl.password = 'password'
) FORMAT PLAIN ENCODE JSON;
```



```
SELECT window_start, window_end, count(*) as events
FROM HOP (github_events, created_at,
         INTERVAL '1 MINUTES', INTERVAL '2 MINUTES')
GROUP BY window_start, window_end
ORDER BY window_start ASC;
```

Databend





```
docker run -p 8000:8000 datafuselabs/databend

go get https://github.com/databendcloud/bend-ingest-kafka

bend-ingest-kafka
  --kafka-bootstrap-servers="127.0.0.1:9092"\
  --kafka-topic="your_topic"\
  --kafka-consumer-group= "Consumer Group"\
  --databend-dsn="http://root:root@127.0.0.1:8000"\
  --databend-table="db1.tbl" \
  --data-format="json" \
  --batch-size=100000 \
  --batch-max-interval=300s
```

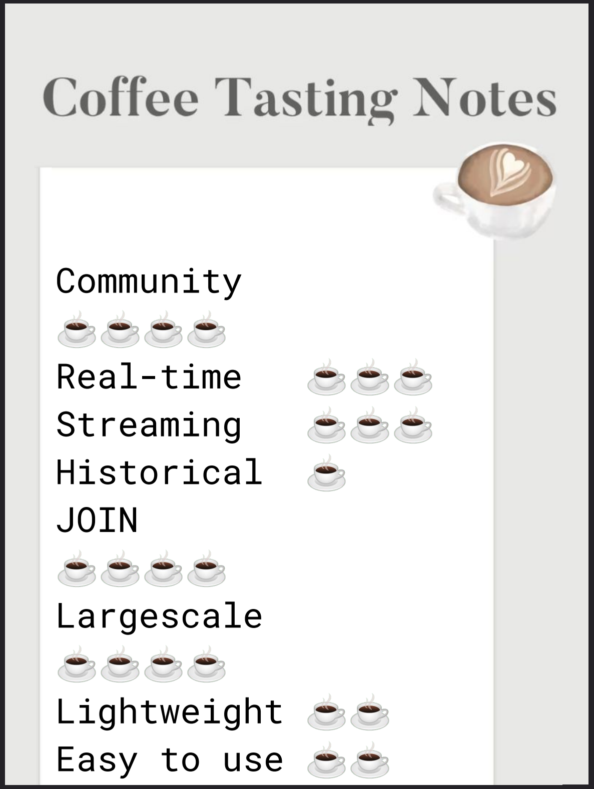
Query Kafka with SQL: Open Source + Cloud + Closed Source

Streaming Processor	Streaming Database	Real-Time Database
Flink, decodable, databricks	ksqlDB, Hazelcast, DeltaStream	Druid, Pinot, Trino
Arroyo, Dozer, Feldera	proton, RisingWave, Materialize	ClickHouse, StarRocks, Rockset, tinybird, Databend, pathway

ksqlDB

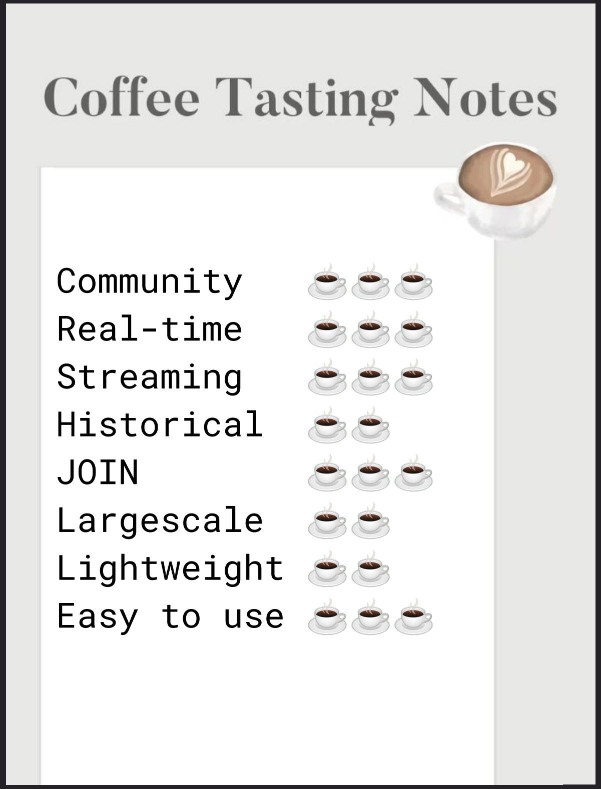
proton

RisingWave



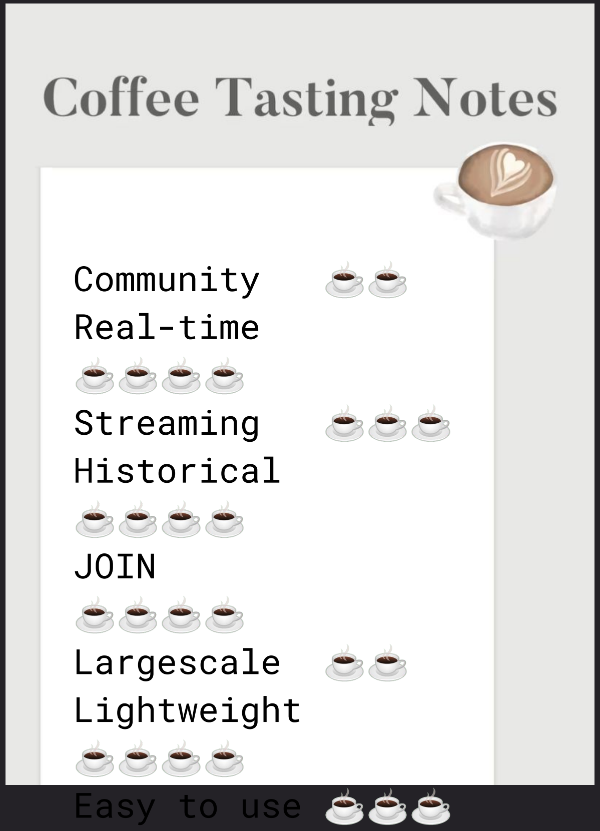
Coffee Tasting Notes

Community	☕☕☕☕
Real-time	☕☕☕
Streaming	☕☕☕
Historical	☕
JOIN	☕☕☕☕
Largescale	☕☕☕☕
Lightweight	☕☕
Easy to use	☕☕



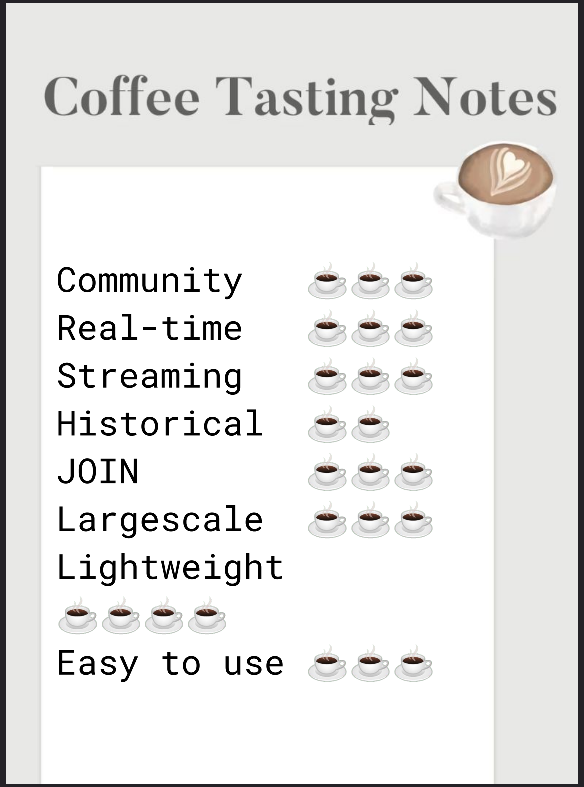
Coffee Tasting Notes

Community	☕☕☕
Real-time	☕☕☕
Streaming	☕☕☕
Historical	☕☕
JOIN	☕☕☕
Largescale	☕☕
Lightweight	☕☕
Easy to use	☕☕☕



Coffee Tasting Notes

Community	☕☕
Real-time	☕☕☕☕
Streaming	☕☕☕
Historical	☕☕☕☕
JOIN	☕☕☕☕
Largescale	☕☕
Lightweight	☕☕☕☕
Easy to use	☕☕☕



Coffee Tasting Notes

Community	☕☕☕
Real-time	☕☕☕
Streaming	☕☕☕
Historical	☕☕
JOIN	☕☕☕
Largescale	☕☕☕
Lightweight	☕☕☕☕
Easy to use	☕☕☕

timeplus | OSA CON 23

Q+A / Thank you!

Try Timeplus Proton (Open Source)
Or sign up for a free cloud account

timeplus.com

Jove Zhong

Gang Tao