Before we start ...

Prepare to Screenshot.

WELCOME!

Thank you for joining us in these unique circumstances. We hope you're safe and well.

EVENT INFORMATION

Ksqldb - a lightweight stream processing engine for Apache Kafka

Mark Teehan

Kafka KRU

How to ask questions and interact during and afterwards

Zoom-chat

Any time during the virtual meetup!

Unmute and ask!

(only before and after talks)

Community slack

channel: #events

Throughout and after the event

THE MEETUP HUB

ALL UPCOMING MEETUPS

NEW MEETUP EMAIL ALERTS

VIDEOS OF PAST MEETUPS

SLIDES FROM THE TALKS

CNFL.IO/MEETUP-HUB

Confluent Community Catalyst program nominations are open for the class of 2020-2021. Find out more and nominate your community heroes!

Confluent.io/Nominate

cnfl.io/slack

MORE LINKS

developer.confluent.io

All the Apache Kafka® learning resources you'll need

cnfl.io/MU-Try-Cloud

Get $50 of free usage every month for your first 3 months!

You are being recorded and this footage may be added to public channels

Confluent are giving new users $50 of free usage per month for their first 3 months

Here's advice on how to use this promotion to try Confluent Cloud for free!

Sign up for a Confluent Cloud account

Please bear in mind that you will be required to enter credit card information but will not be charged unless you go over the $50 usage in any of the first 3 months or if you don't cancel your subscription before the end of your promotion.

You won't be charged if you don't go over the limit!

Get the benefits of Confluent Cloud, but keep an eye on your your account making sure that you have enough remaining free credits available for the rest of your subscription month!!

Cancel before the 3 months end If you don't want to continue past the promotion

If you fail to cancel within your first three months you will start being charged full price. To cancel, immediately stop all streaming and storing data in Confluent Cloud and email cloud-support@confluent.io

bit.ly/**TryConfluentCloud**

Available on

Google Cloud Platform

Microsoft Azure

aws

KAFKA SUMMIT 2020

Event Streaming Everywhere • Aug 24 - 25

Cnfl.io/Summit-2020-MC

THE VIRTUAL CONFERENCE BY

CONFLUENT

CONFLUENT

Project Metamorphosis

Unveiling the next-gen event streaming platform

Jay Kreps

Co-founder and CEO

Confluent

July 1

(yesterday!)

cnfl.io/pm

cnfl.io/MU-Try-Cloud

Lightweight stream processing engine

For Apache Kafka

ksqlDB

THE

FOUR-CLUSTER

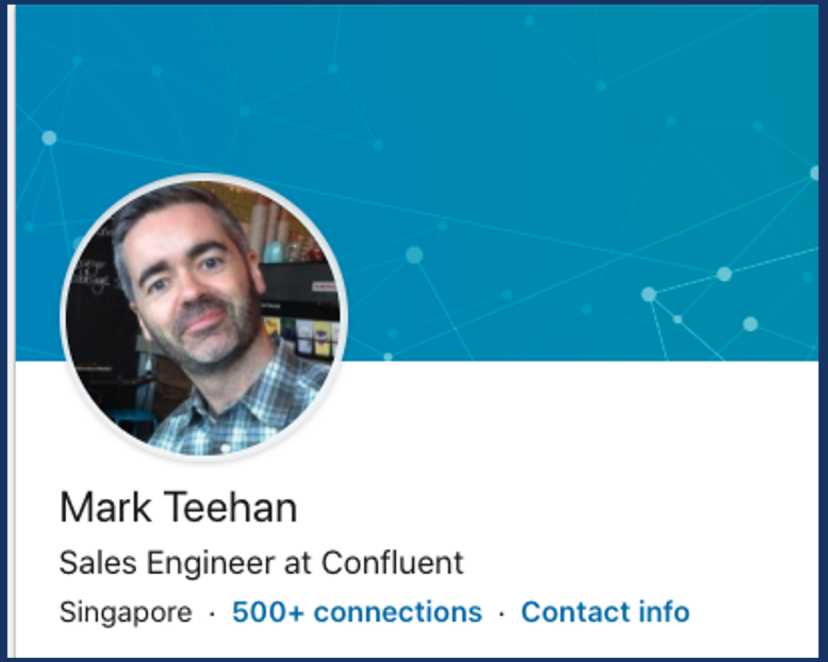
PROBLEM

Mark Teehan
Solution Engineer
Confluent
(Singapore)

Translation:

이준호 (Brandon/JUNHO LEE)

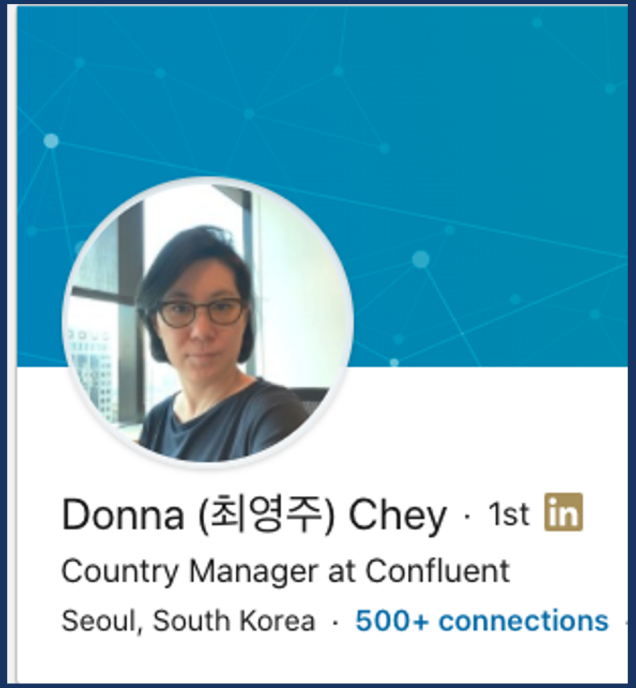
Kafka KRU - Oct 25 2019



kafka KRU

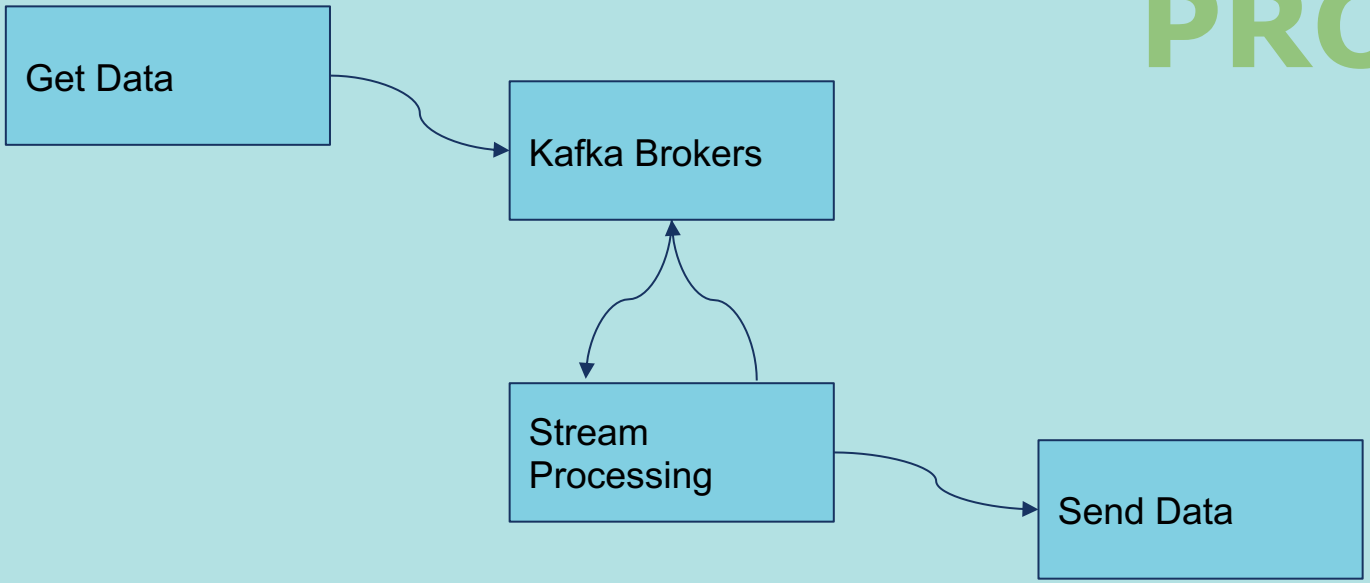
teehan@confluent.io

Confluent in Korea!
최영주 한국영업대표

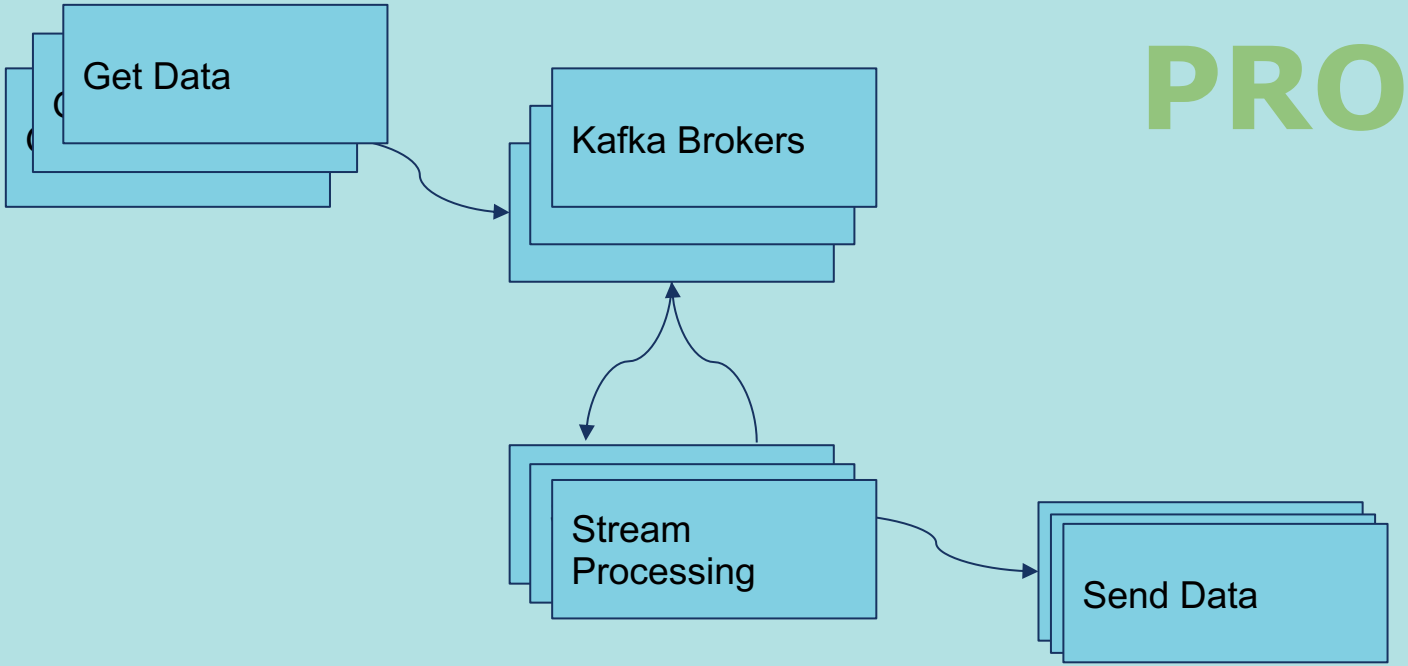


dchey@confluent.io

THE FOUR-CLUSTER PROBLEM



THE FOUR-CLUSTER PROBLEM



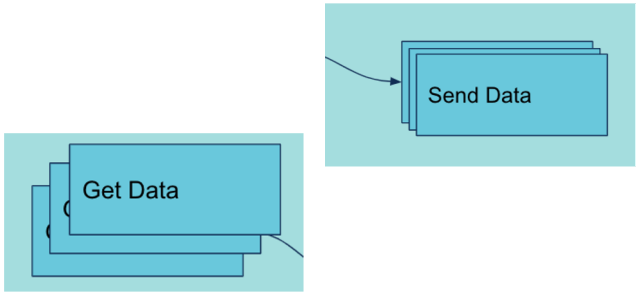
What is ksqlDB?

ksqlDB is an **event streaming database** purpose-built to help developers create **stream processing** applications on top of Apache Kafka®.

Is ksqlDB owned by the Apache Software Foundation?

No. This is a community component of Confluent Platform. ksqlDB is owned and maintained by Confluent Inc. as part of its Confluent Platform product. However, ksqlDB is licensed under the Confluent Community License.

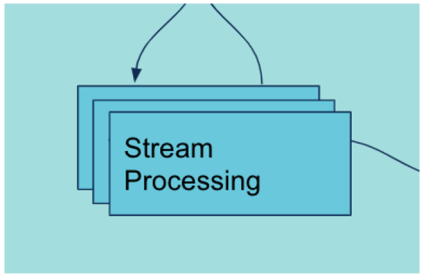
How does ksqlDB solve the four-cluster-problem?



ksqlDB natively integrates with Connect by either

1.communicating with an external Connect cluster or

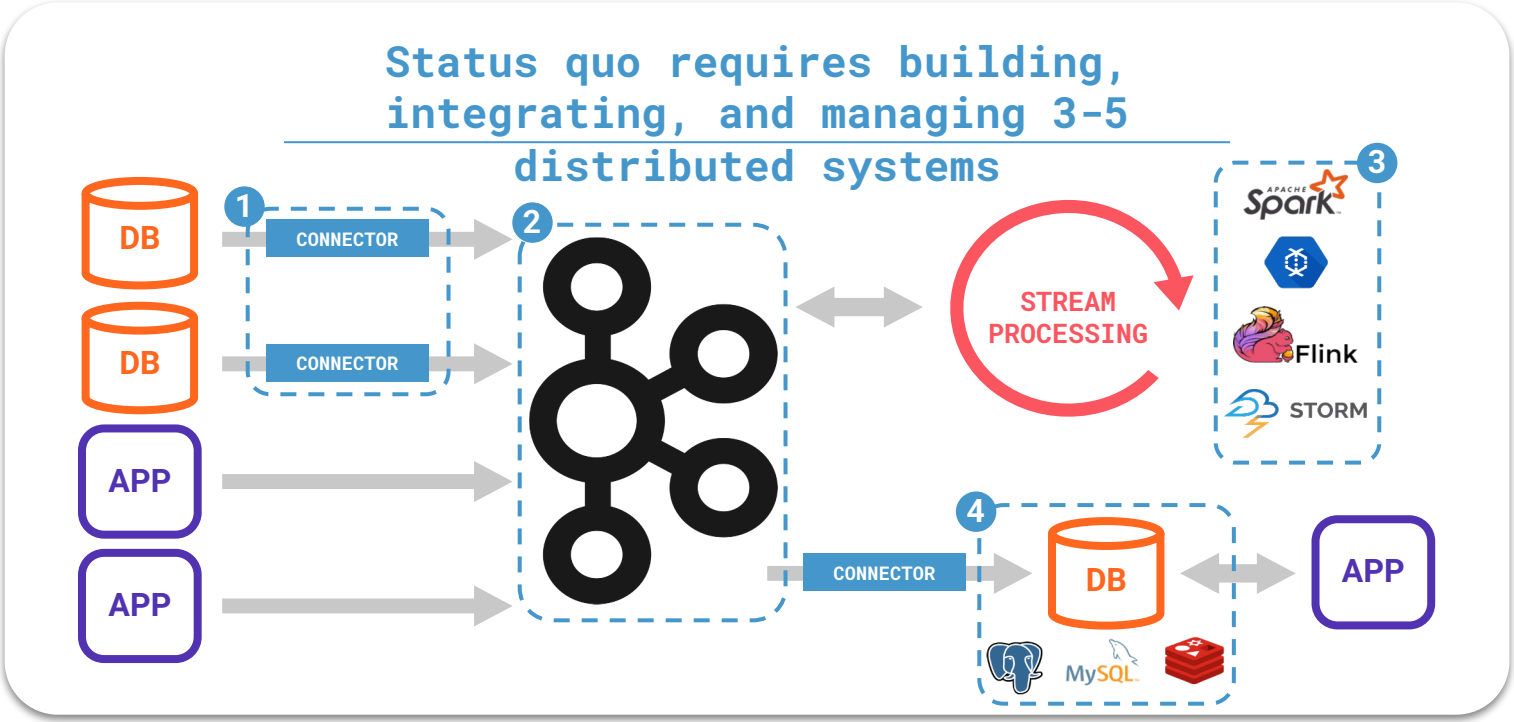
2. running Connect embedded within the KSQL server process.



ksqlDB is an event streaming database purpose-built to help developers create stream processing applications on top of Apache Kafka®.

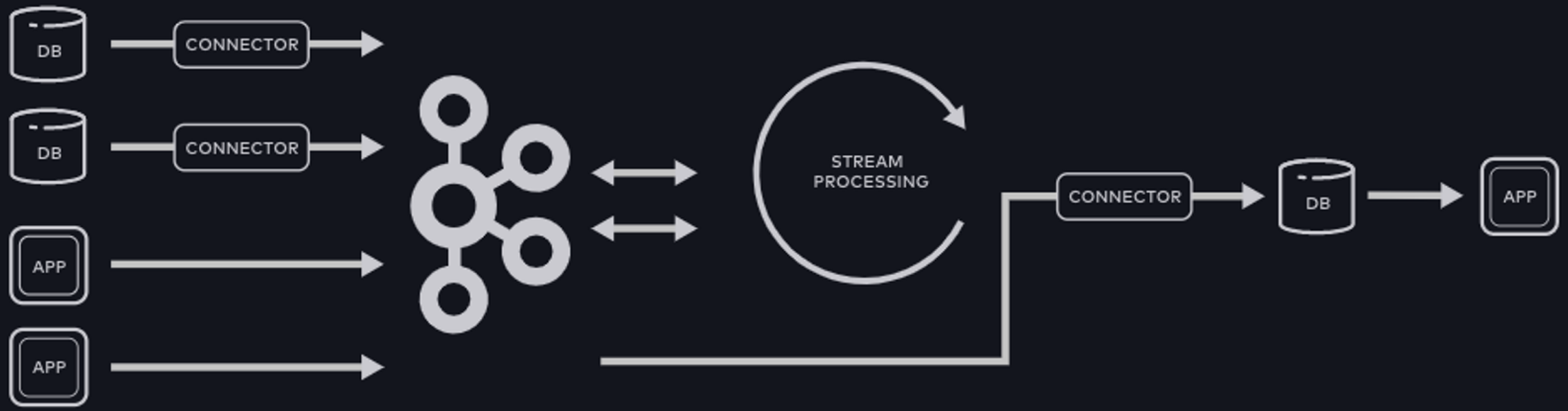
1. Stream Processing with *STREAMs and TABLEs*
2. Materialized Views, backed by RocksDB
3. *Push* and *Pull* Queries
4. Join Event Streams

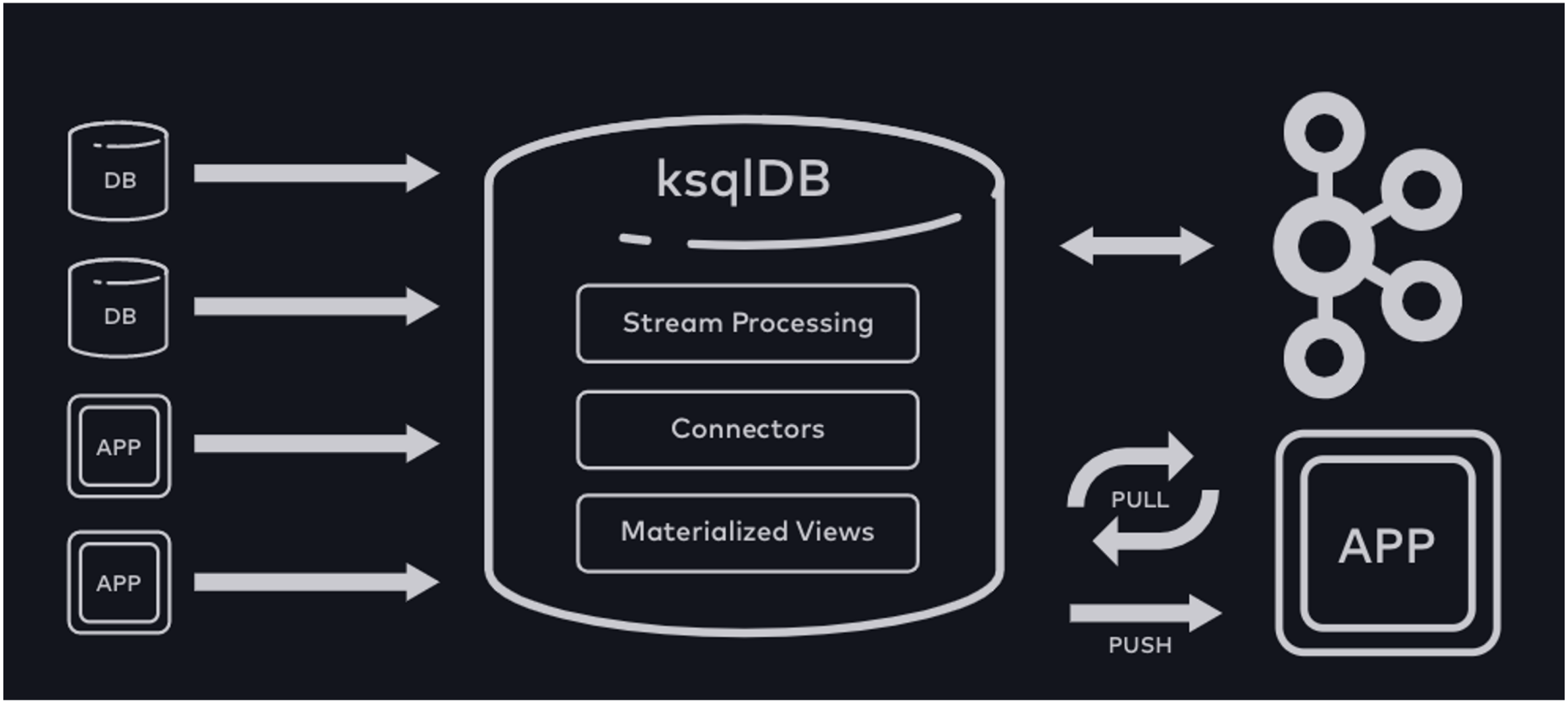
Stream processing architectures can be complex



One mental model for the entire stack.

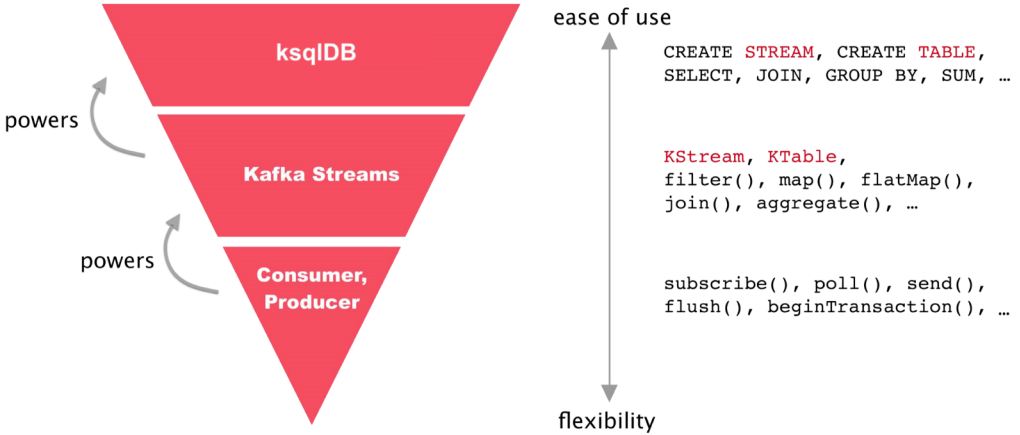
Today, nearly all streaming architectures are complex, piecemeal solutions. They comprise multiple subsystems, each with its own mental model.




DB
DB
APP
APP
ksqlDB
Stream Processing
Connectors
Materialized Views
PULL
PUSH
APP

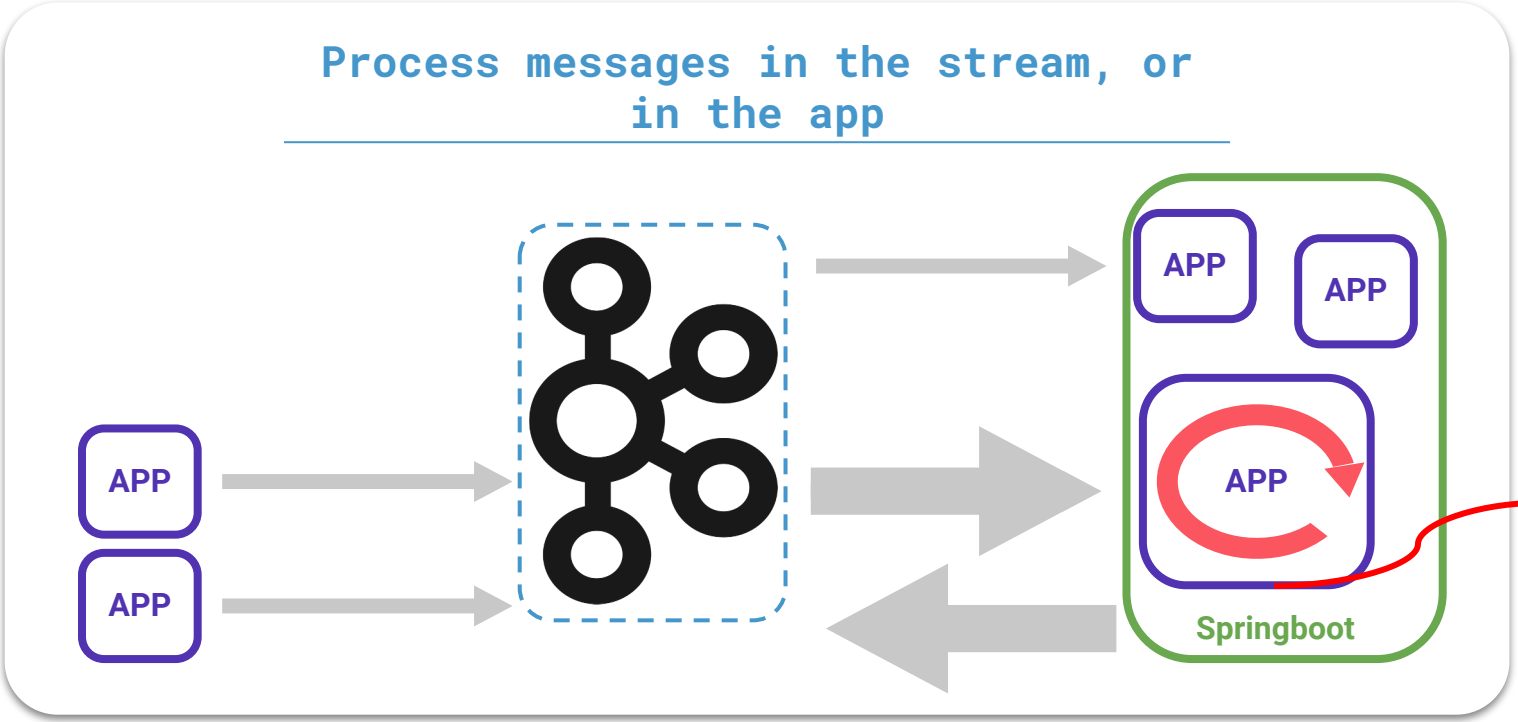
Relationship to Kafka Streams

- ksqlDB is the streaming database for Apache Kafka®.
- With ksqlDB, you can write event streaming applications by using a lightweight SQL syntax.
- Kafka Streams is the Kafka library for writing streaming applications and microservices in Java and Scala.
- ksqlDB is built on Kafka Streams, a robust stream processing framework that is part of Kafka.

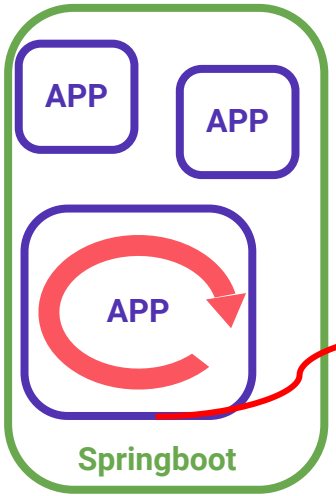


“We don’t do stream processing…”

Are all of your apps equal?



Only consume what you need



It is common to ignore parts of messages

Message: {col1, col2, col3, col4, col5, col6, col7, col8, col9}

Message: {col1, col2, col3, col4, col5, col6, col7, col8, col9}

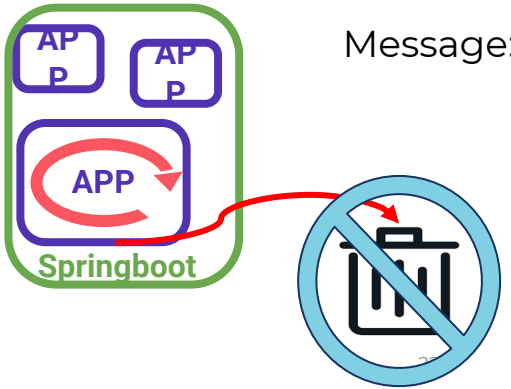
Filter to what the consumer needs:

Message: {col1, col2, col3,col8}

Filtering Columns is stream processing

Filter to what the consumer needs

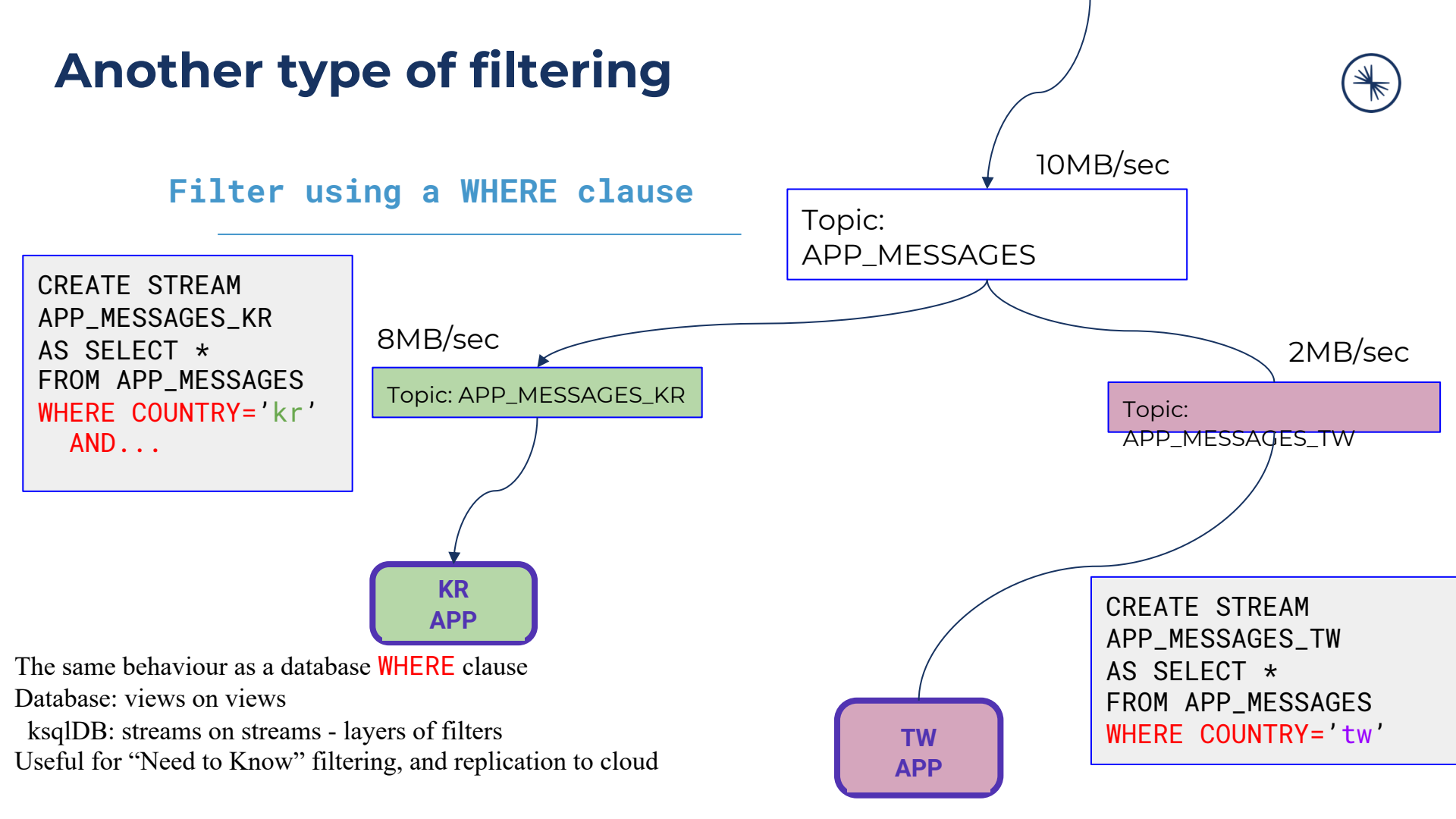
Message: {col1, col2, col3,col8}




```
CREATE STREAM
STR_MEGA_APP_MESSAGES AS
SELECT col1
      ,col2
      ,col3
      ,col8
  FROM FROM APP_MESSAGES;
```


Examples of large messages:
ISO Payment Messages (1200 attributes)
Insurance Policies
Syslog entries

Another type of filtering



The same behaviour as a database WHERE clause
Database: views on views
ksqlDB: streams on streams - layers of filters
Useful for “Need to Know” filtering, and replication to cloud

Stream processing:

Filtering (reduce message size)

Filtering (WHERE clause)

JOINs

Aggregation (GROUP BY)

ksqlDB *Collections:* Streams and Tables

STREAM - a regular Kafka Topic - append only

TABLE - a compacted Kafka Topic - uses tombstones

Just normal topics: offsets, partitions, key+value, schemas, consumer groups etc

Use *INSERT INTO..VALUES* to Insert/Produce into a STREAM or TABLE

You can declare a STREAM or TABLE over an existing topic, or you can create a new topic for the collection at declaration time

Joins

Just like a database join - merge streams from multiple topics

The topics being joined must have the same configuration for partitioning

```
STREAM+STREAM=STREAM

TABLE+TABLE   =TABLE

STREAM+TABLE  =STREAM
```

For STREAM+STREAM joins you must specify a WITHIN clause for matching records within a specified time interval.

A *WITHIN* clause also manages out-of-order and late-arriving data

"...WITHIN 60 MINUTES…"

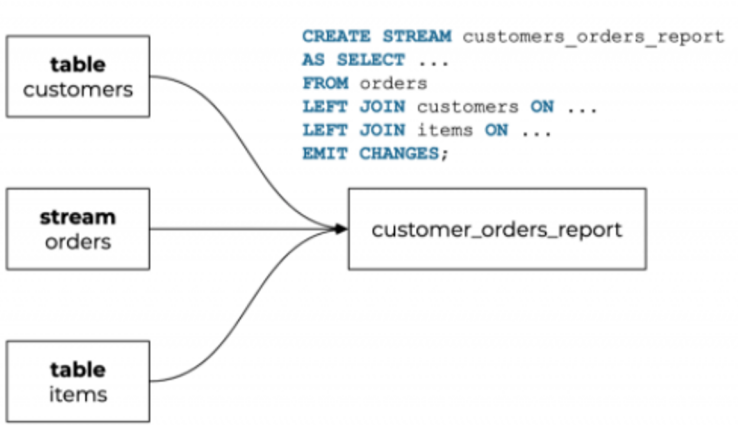
Support for *Multi-Join*

Ksqldb 0.9.0 (19-May-2020): You can do multiple joins into a single statement.

There is no limit to the number of joins in a single statement.

This works with all join types: inner, left, and outer joins.

Normal restrictions apply: you can't repartition tables



ksqlDB Change Log

https://github.com/confluentinc/ksql/blob/master/CHANGELOG.md

Change Log

0.10.0 (2020-06-25)

Features

- Any key name (#5093) (1f0ca3e)
- Explicit keys (#5533) (d0db0cf)
- add extra log messages for pull queries (#4909) (d622ecc)
- Adds the ability have internal endpoints listen on ksql.internal.listener (#5212) (46acb73)
- create MetricsContext for ksql metrics reporters (#5528) (50561a5)
- drop WITH(KEY) syntax (#5363) (bb43d23)
- expose JMX metric that classifies an error state (#5374) (52271bf)
- Expose Vert.x metrics (#5340) (e82f762)
- introduce RegexClassifier for classifying errors via cfg (#5412) (b25dd98)
- Pull Queries: QPS check utilizes internal API flag to determine if forwarded (#5392) (08b428f)
- reload TLS certificate without restarting server (#5516) (a5920b0)
- support TIMESTAMP being a key column (#5542) (286ce08)
- turn on snappy compression for produced data (#5495) (27d8ad5)
- **client:** Java client with push + pull query support (#5200) (280ef0c)

ksqlDB queries

`PULL Queries` - look up information at a point in time (one result)

`PUSH Queries` - subscribe to a result as it changes in real-time (a continuous stream)
"push to the user"

```
PULL: Request/Response

PUSH: Asynchronous
flows
```

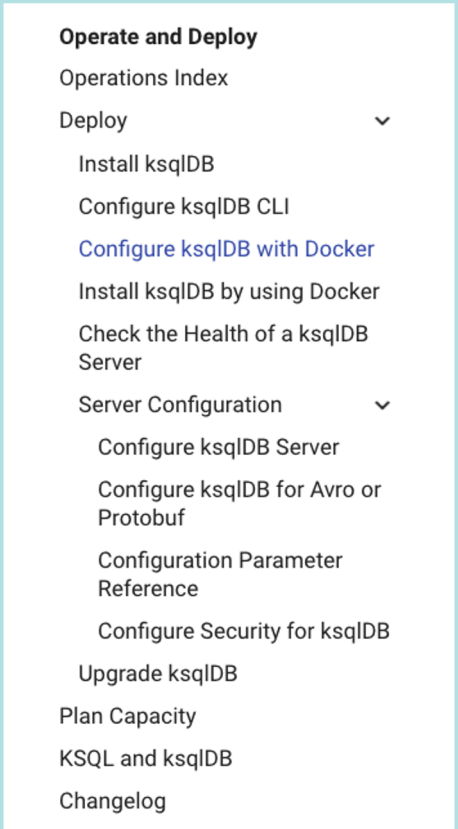
```
PULL:

SELECT LAT, LONG
  FROM LOCATION
WHERE ROWKEY='SGX1191'
```

```
PUSH:

SELECT *
  FROM PAGEVIEWS
  EMIT CHANGES
 LIMIT 100;
```

Running ksqlDB




```yaml
ksqldb:
  image: confluentinc/ksqldb-server:0.9.0
  hostname: p35-ksqldb
  container_name: p35-ksqldb
  depends_on:
    - kafka1
    - schema-registry
    - kafka-connect
  ports:
    - 8088:8088
  environment:
    KSQL_BOOTSTRAP_SERVERS: p35-kafka1:29092
    KSQL_LISTENERS: http://p35-ksqldb:8088
    KSQL_KSQL_LOGGING_PROCESSING_STREAM_AUTO_CREATE: "true"
    KSQL_KSQL_LOGGING_PROCESSING_TOPIC_AUTO_CREATE: "true"
    KSQL_KSQL_CONNECT_URL: http://p35-kc:8083
    KSQL_KSQL_SCHEMA_REGISTRY_URL: http://p35-sr:8081
    KSQL_KSQL_SERVICE_ID: p35
    KSQL_KSQL_INTERNAL_TOPIC_REPLICAS: 1
```

Processing Guarantees

ksqlDB supports

- **At-least-once semantics**
  Records are never lost but may be redelivered. If your stream processing application fails, no data records are lost and fail to be processed, but some data records may be re-read and therefore re-processed. At-least-once semantics is **enabled by default** in your ksqlDB configuration, with processing.guarantee="at_least_once".
- **Exactly-once semantics**
  Records are processed once. If a producer within a ksqlDB application sends a duplicate record, it's written to the broker exactly once.
  Exactly-once stream processing is the ability to execute a read-process-write operation exactly one time.
  To enable exactly-once semantics, set processing.guarantee="exactly_once" in your ksqlDB configuration.
- **Important**: Use the exactly_once setting with care. To achieve a true exactly-once system, end consumers and producers must also implement exactly-once semantics.

Q&A

Demo

Confluent in Korea!
최영주 한국영업대표

Watch out for a free 2-hour ksqlDB technical virtual hands-on Workshop later in 2020.

Free of charge, but places are limited. Please register interest early.
Email: dchey@confluent.io

Or contact: 고승범 kafka KRU