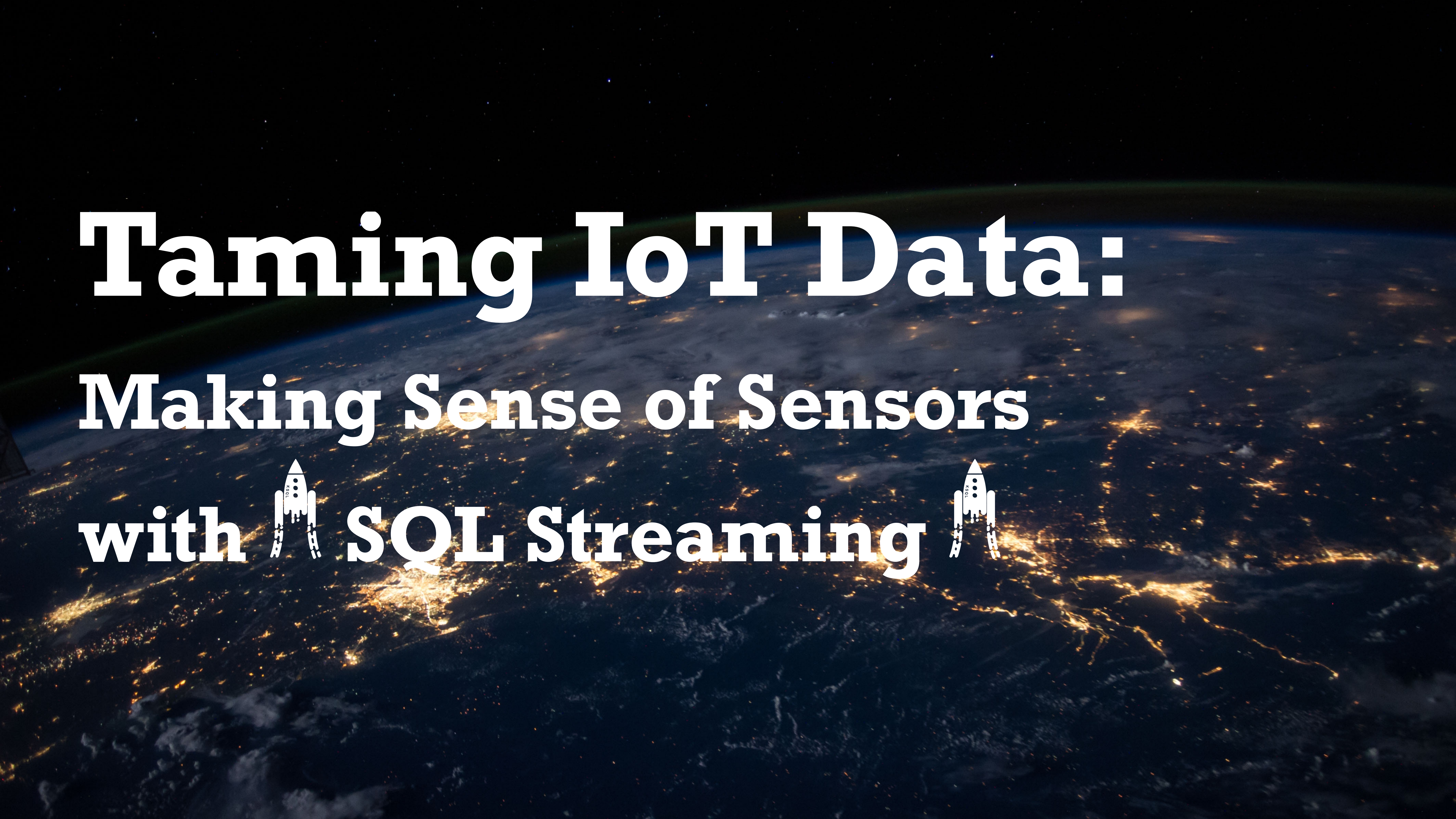


Taming IoT Data:

Making Sense of Sensors with 🚀 SQL Streaming 🚀

$ whoami

- Hans-Peter Grahsl
- working & living in Graz 🇦🇹
- technical trainer at NETCONOMY
- independent consultant & engineer
- associate lecturer
- 🗣 irregular conference speaker 📢

WHAT IS STREAMING

🤔 ❓ 🤔 ❓ 🤔 ❓ 🤔

"a type of **data processing** that is designed with **infinite data sets** in mind"

— Tyler Akidau

Streaming == BIG DEAL

1. **unbounded data sets are prevalent**
   ➡ never-ending data streams need purpose-built systems

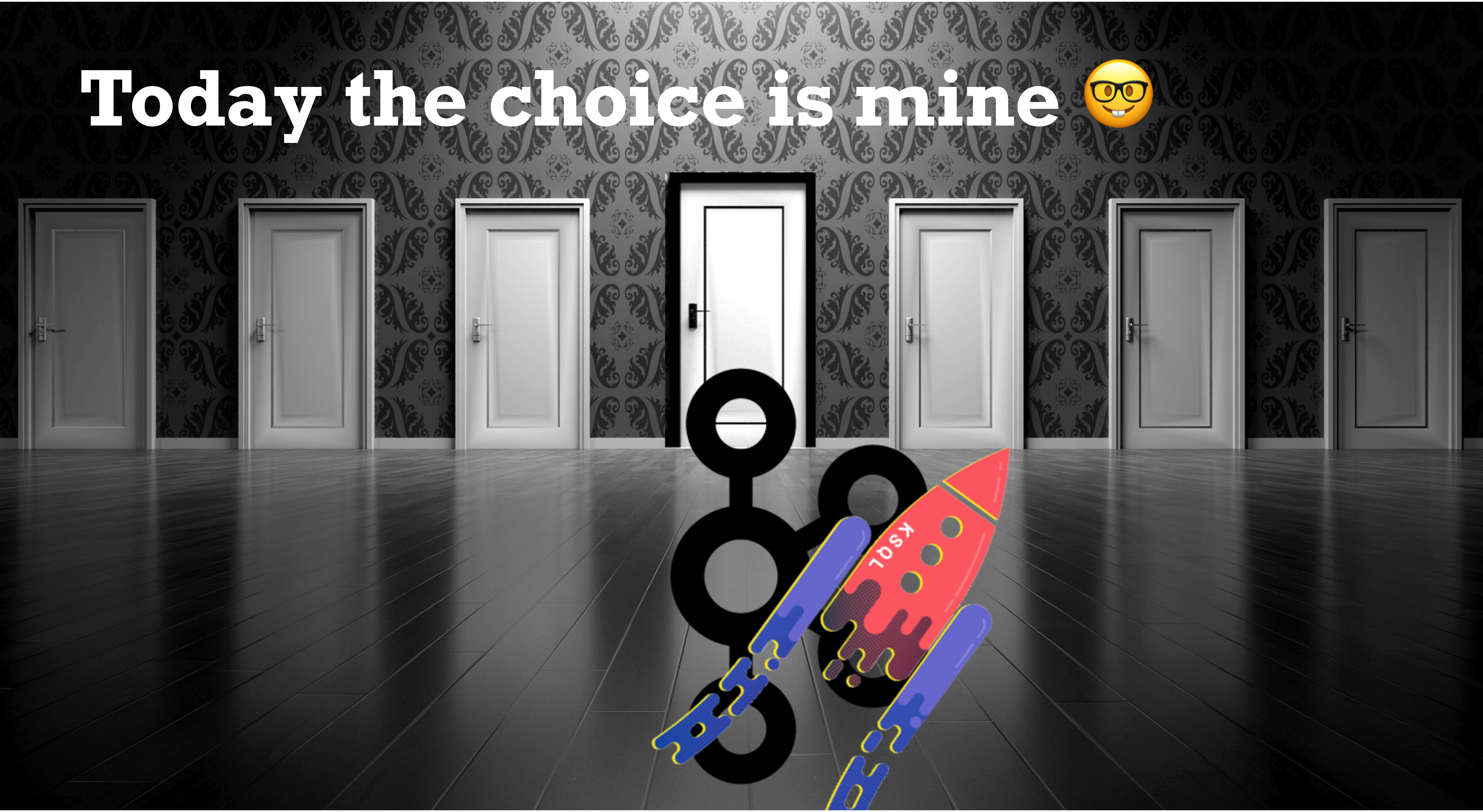
2. **people crave for timely information**
   ➡ stream processing technology aids lower latencies

BIGGEST Challenge?

These and many many more…

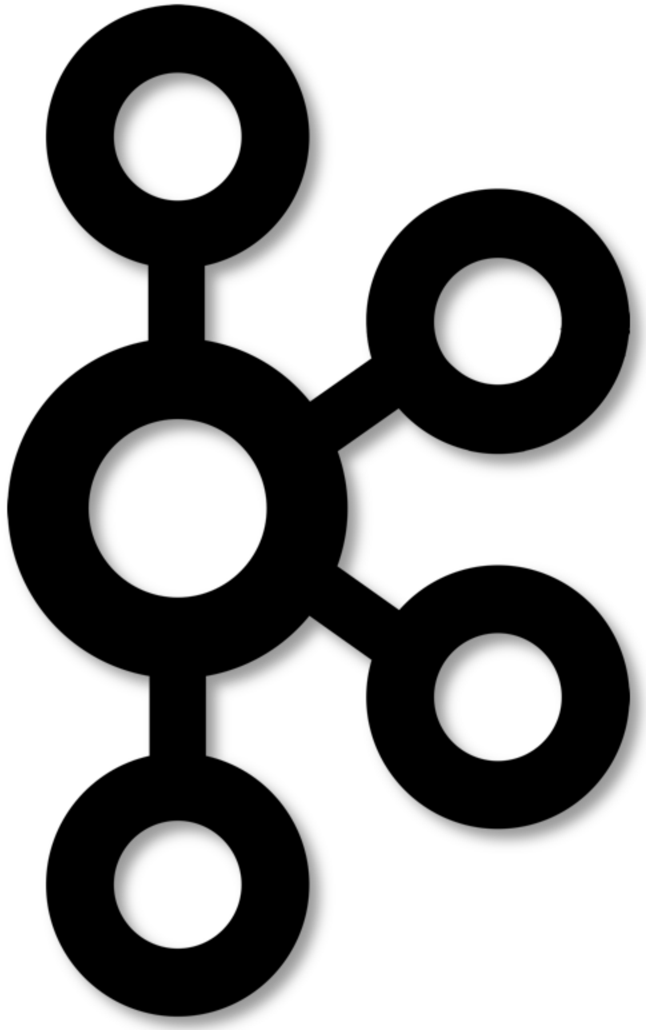
samza
APACHE Spark
KSQL
APEX
beam

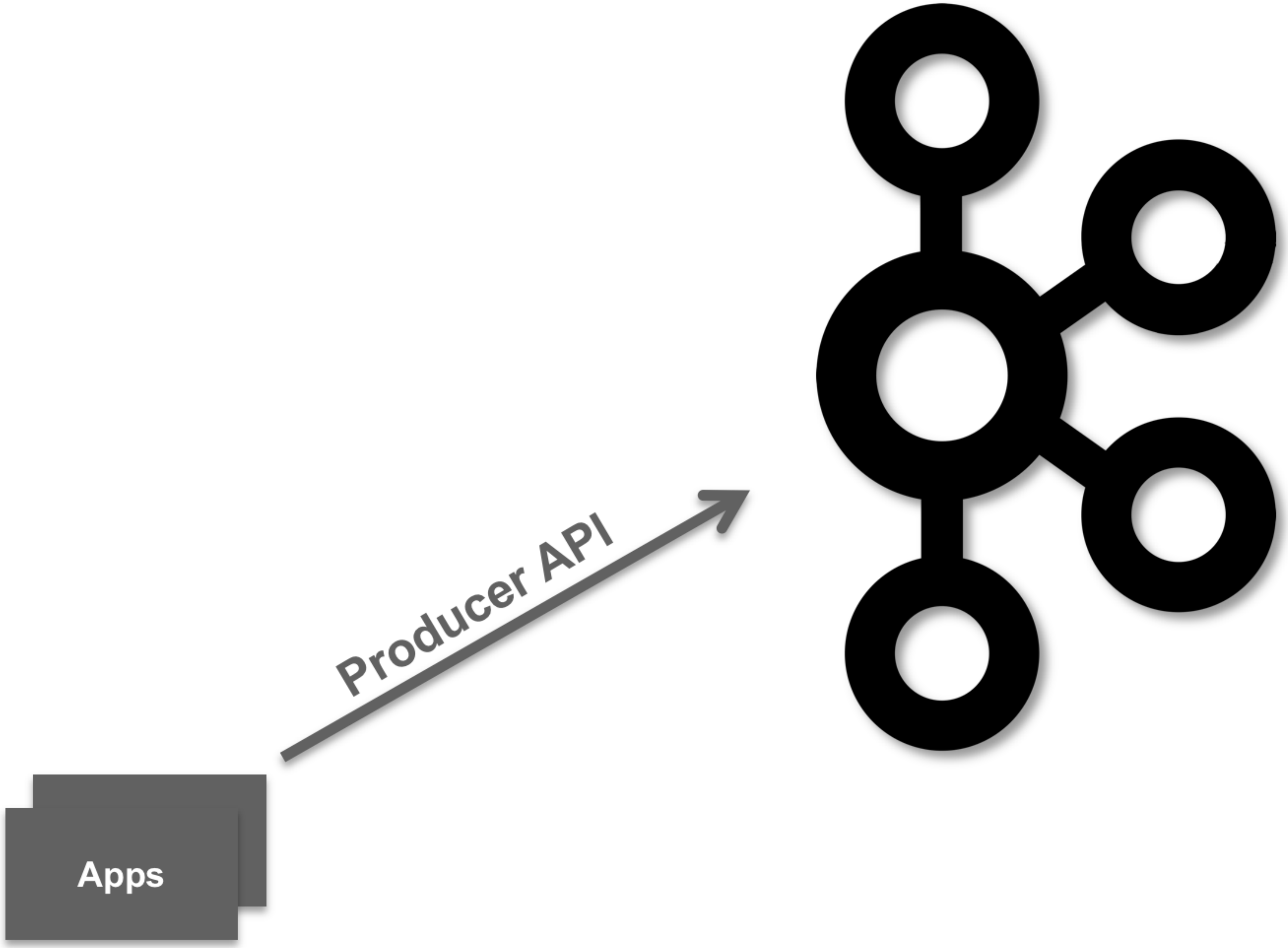
Today the choice is mine 🤓



Apache Kafka

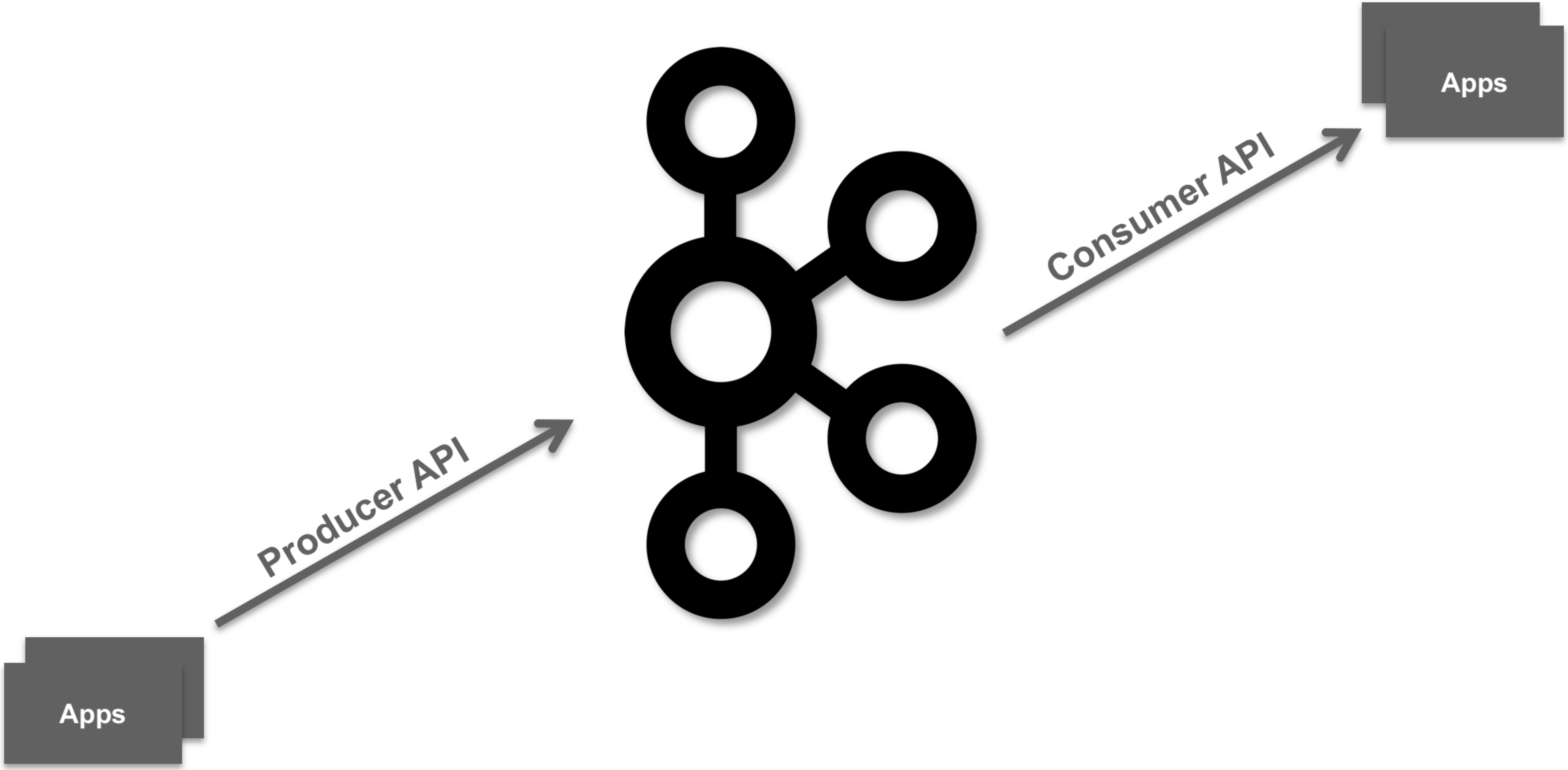
STREAMING PLATFORM

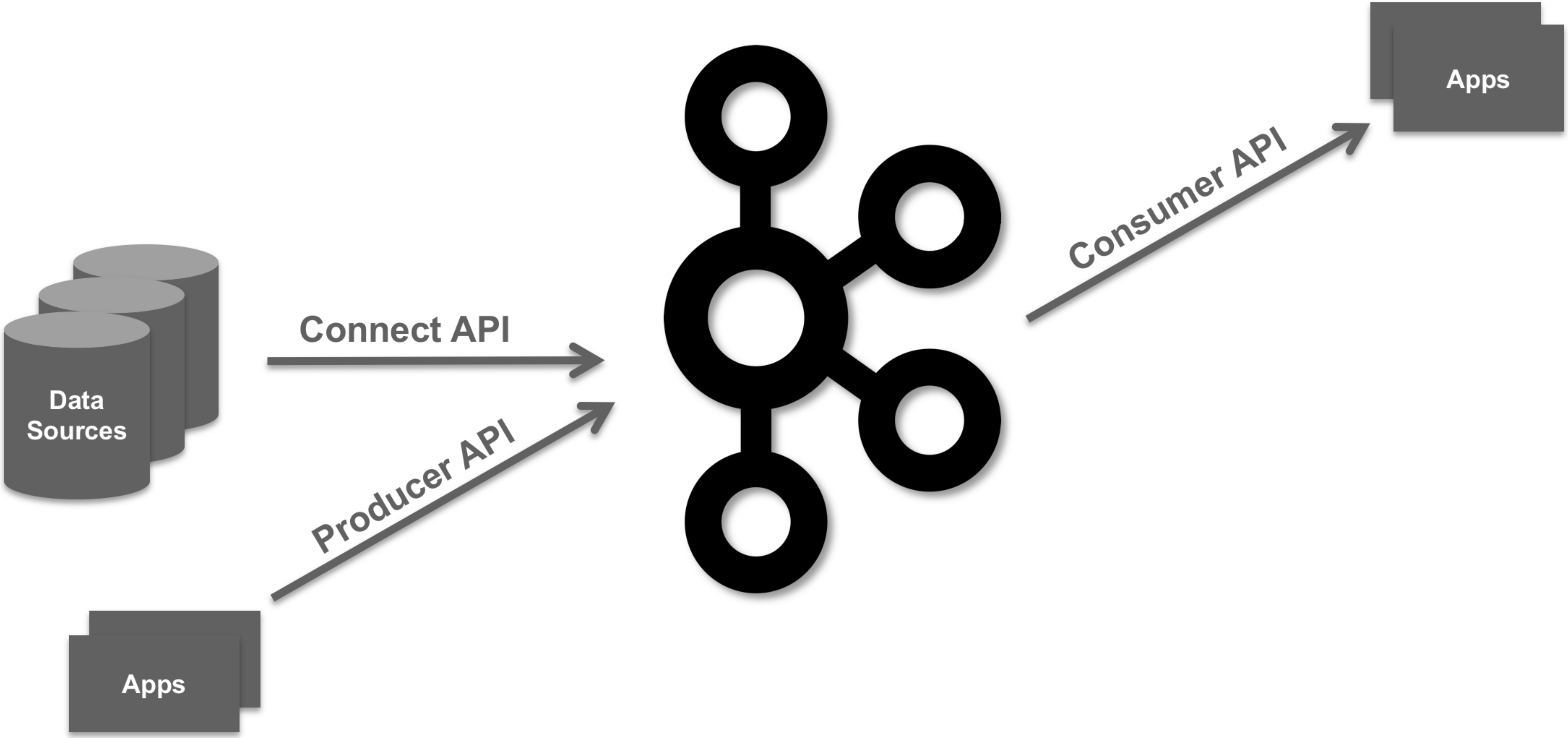


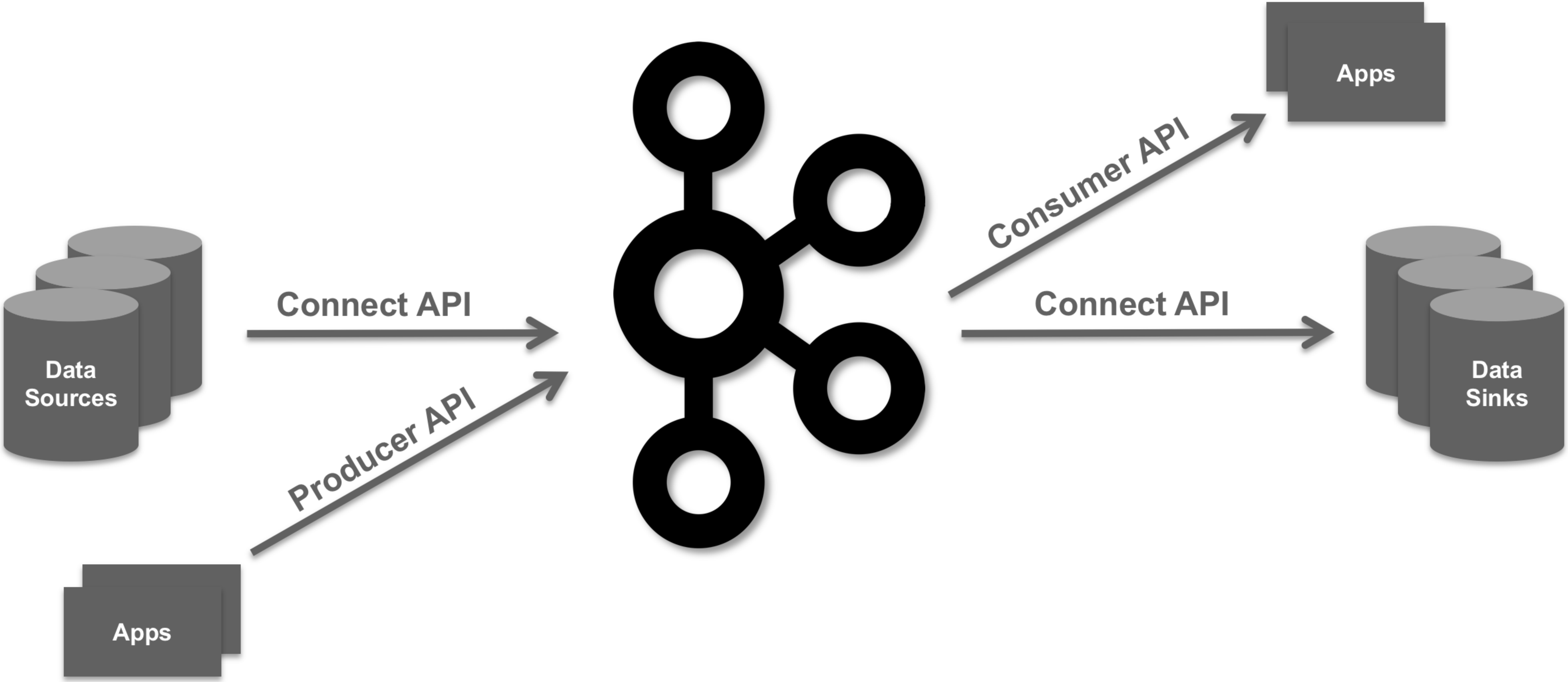


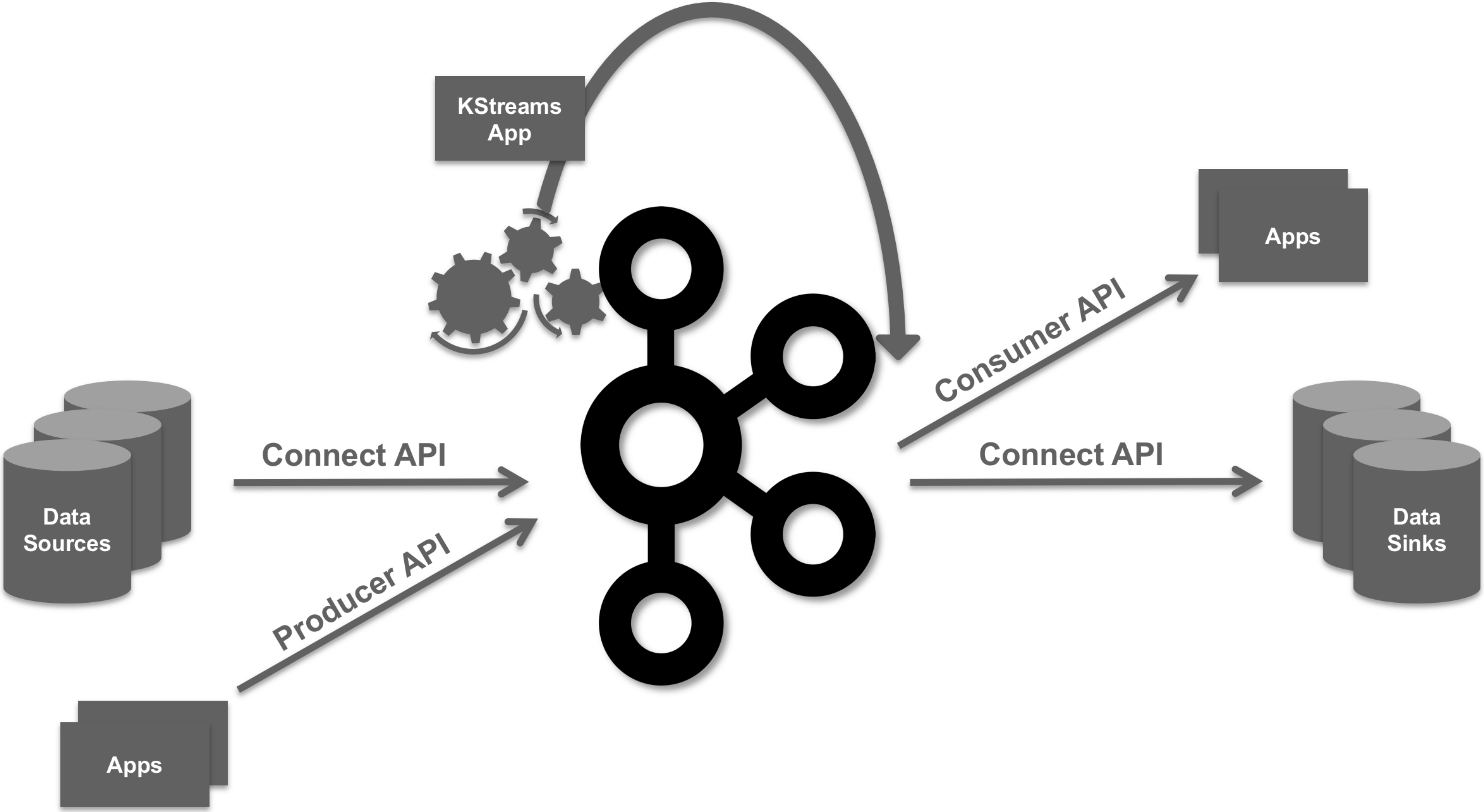
Producer API

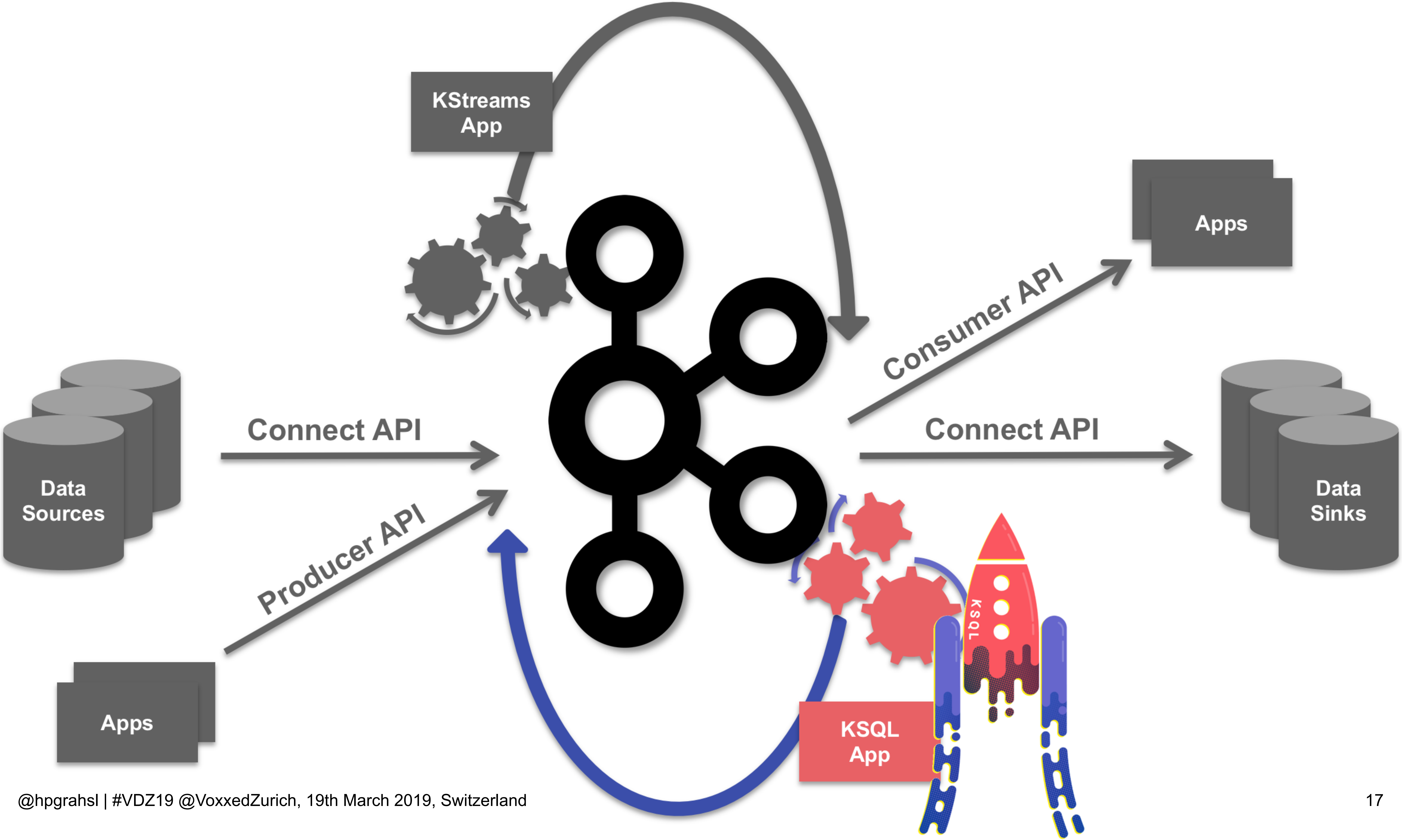
Apps

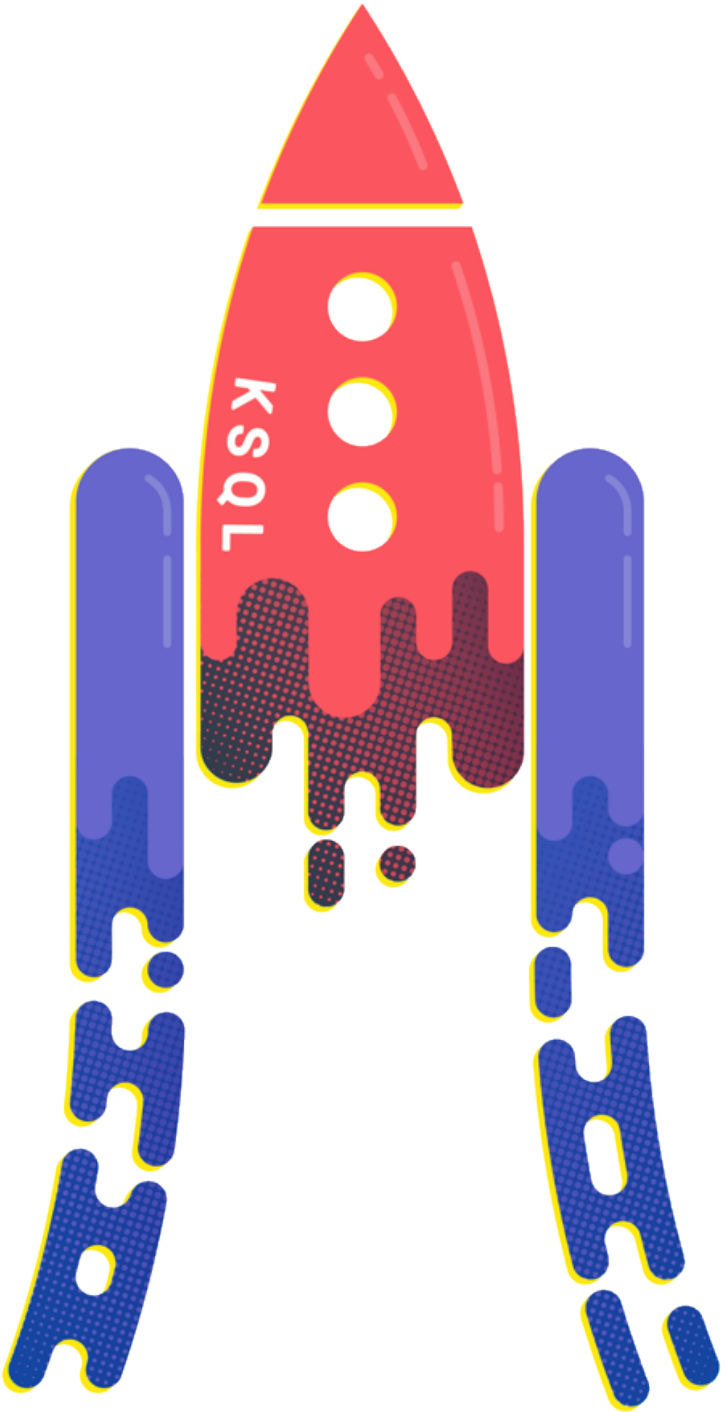

Apps
Producer API
Consumer API
Apps


Apps
Consumer API
Connect API
Data Sources
Producer API
Apps

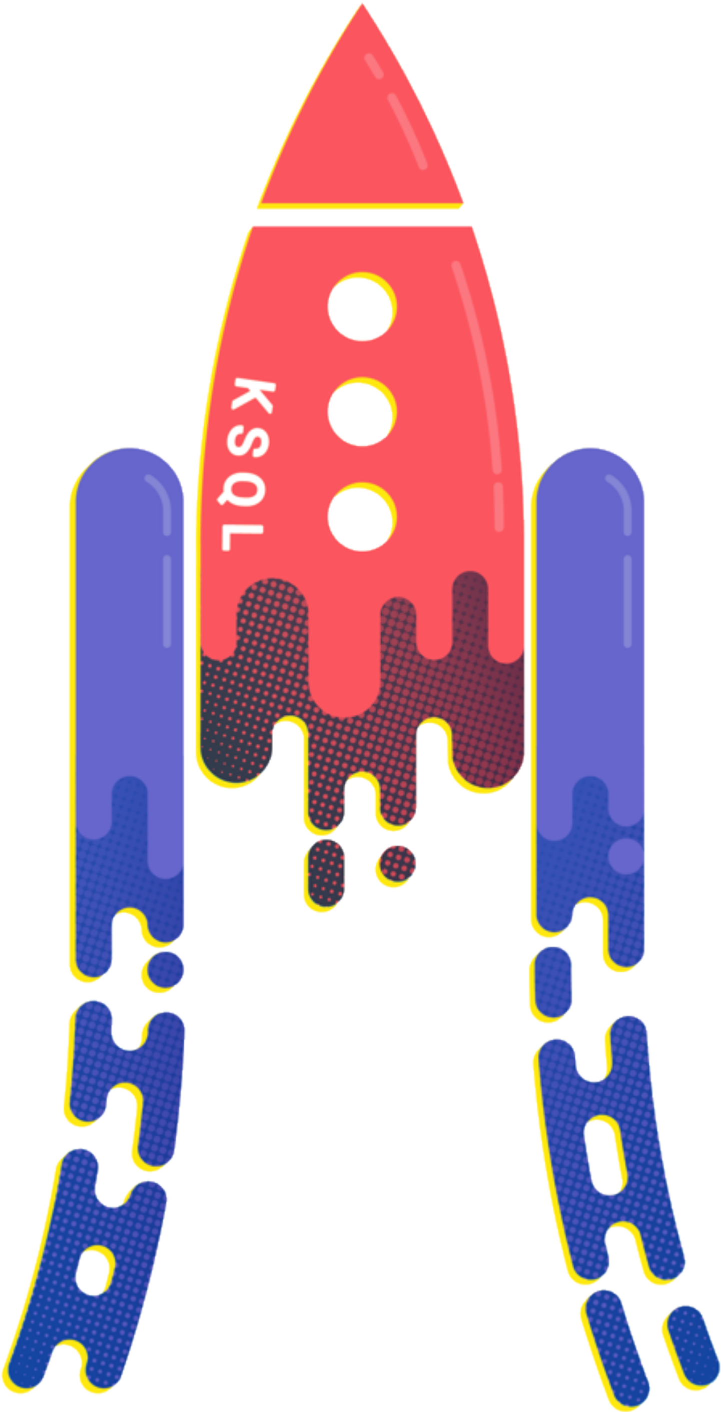

Data
Sources
Connect API
Producer API
Apps
Consumer API
Apps
Connect API
Data
Sinks


KStreams App
Apps
Consumer API
Data Sources
Connect API
Connect API
Data Sinks
Producer API
Apps


KStreams App
Apps
Consumer API
Connect API
Data Sinks
Data Sources
Connect API
Producer API
Apps
KSQL
KSQL App

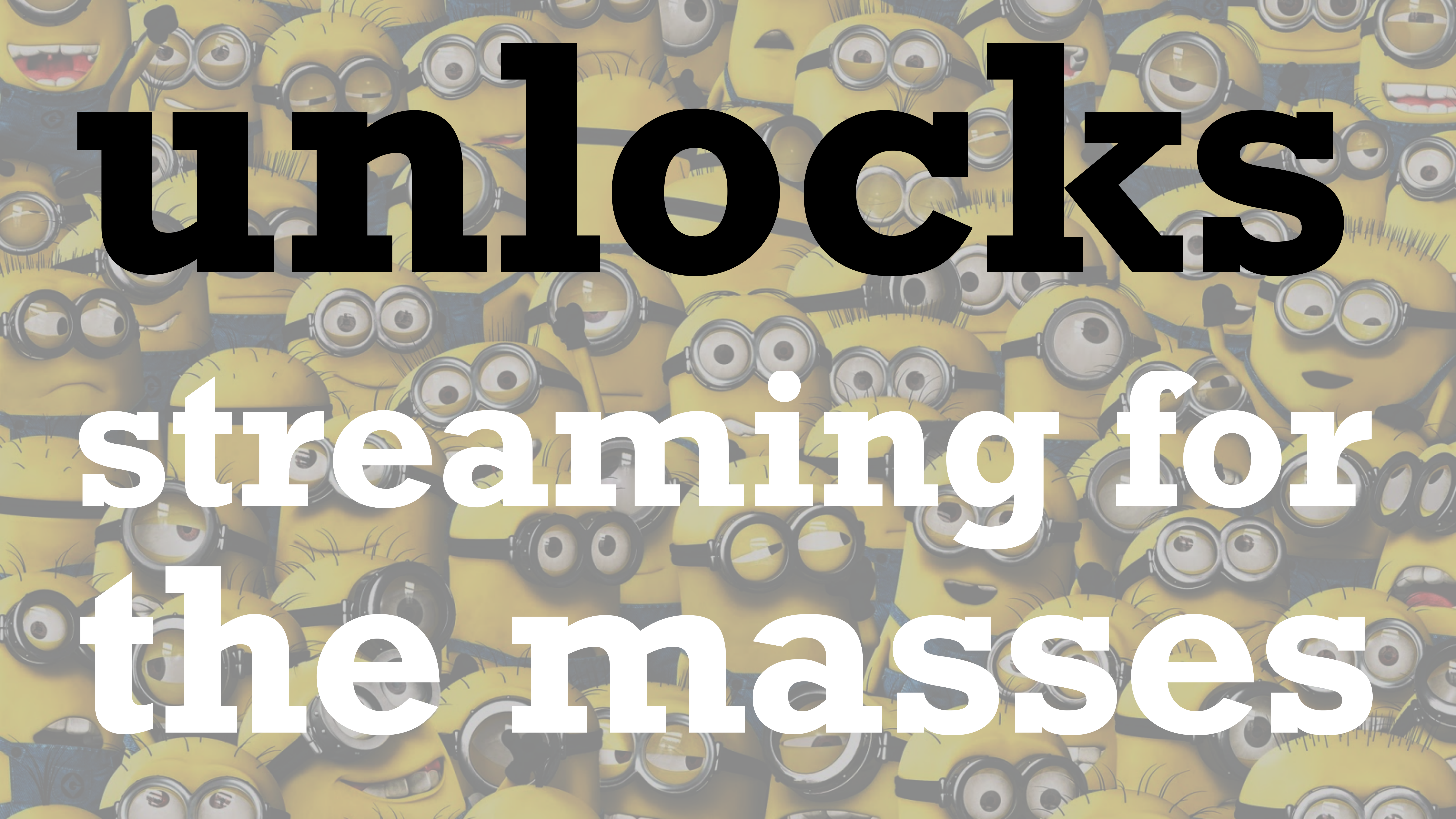


Kafka's streaming SQL engine



declarative stream processing language

skyrocketing developer productivity



unlocks

streaming for the masses

KSQL's Nature

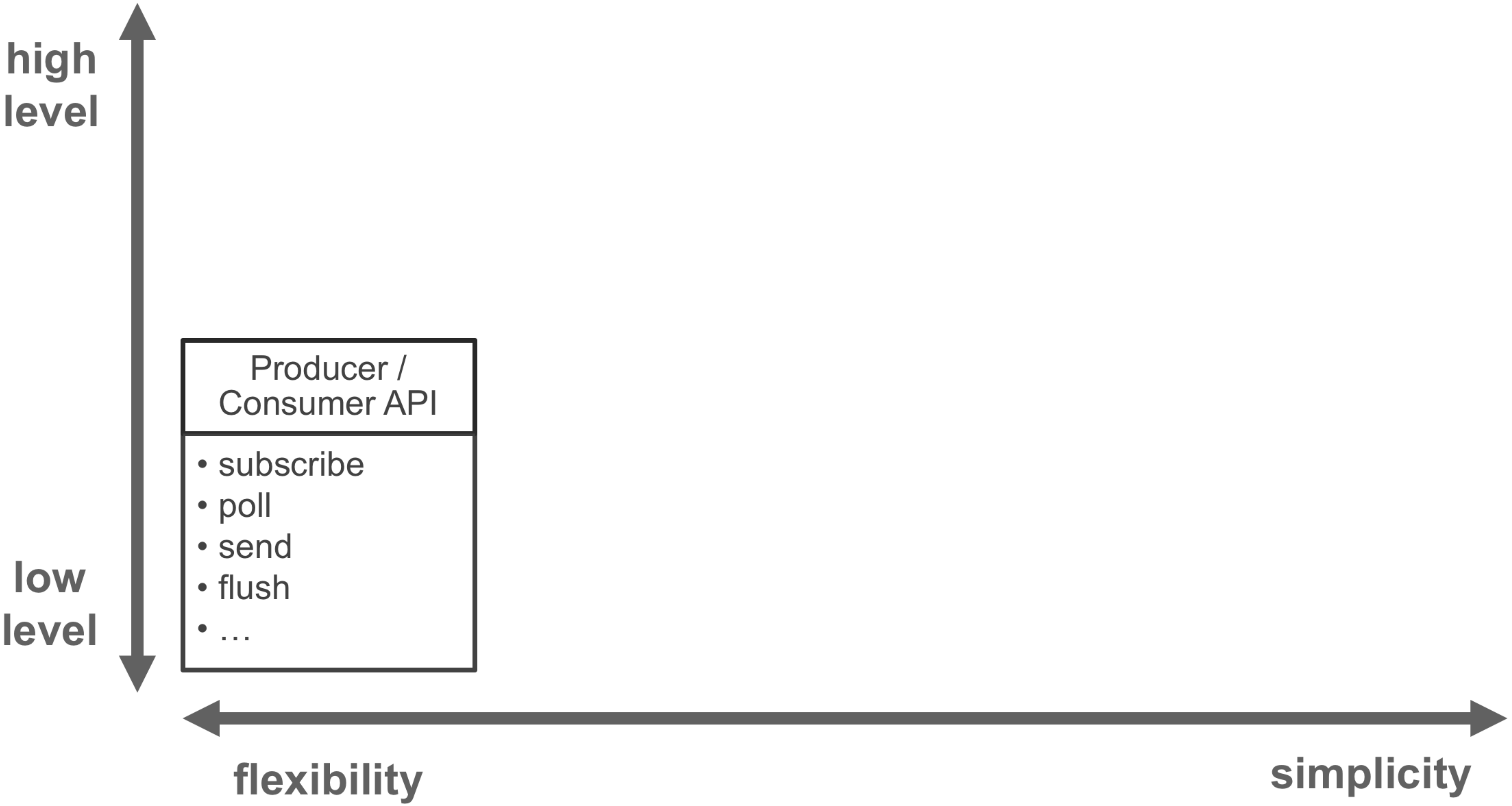
- built on top of **Kafka Streams**
- **SQL only** (not embedded)
- **NO(!) coding skills** required
- extremely **low entry barrier**
  - familiar syntax and semantics
  - concise and expressive
- **joins, aggregations, windowing**
- **UD(A)Fs** and *UDTFs coming soon...*

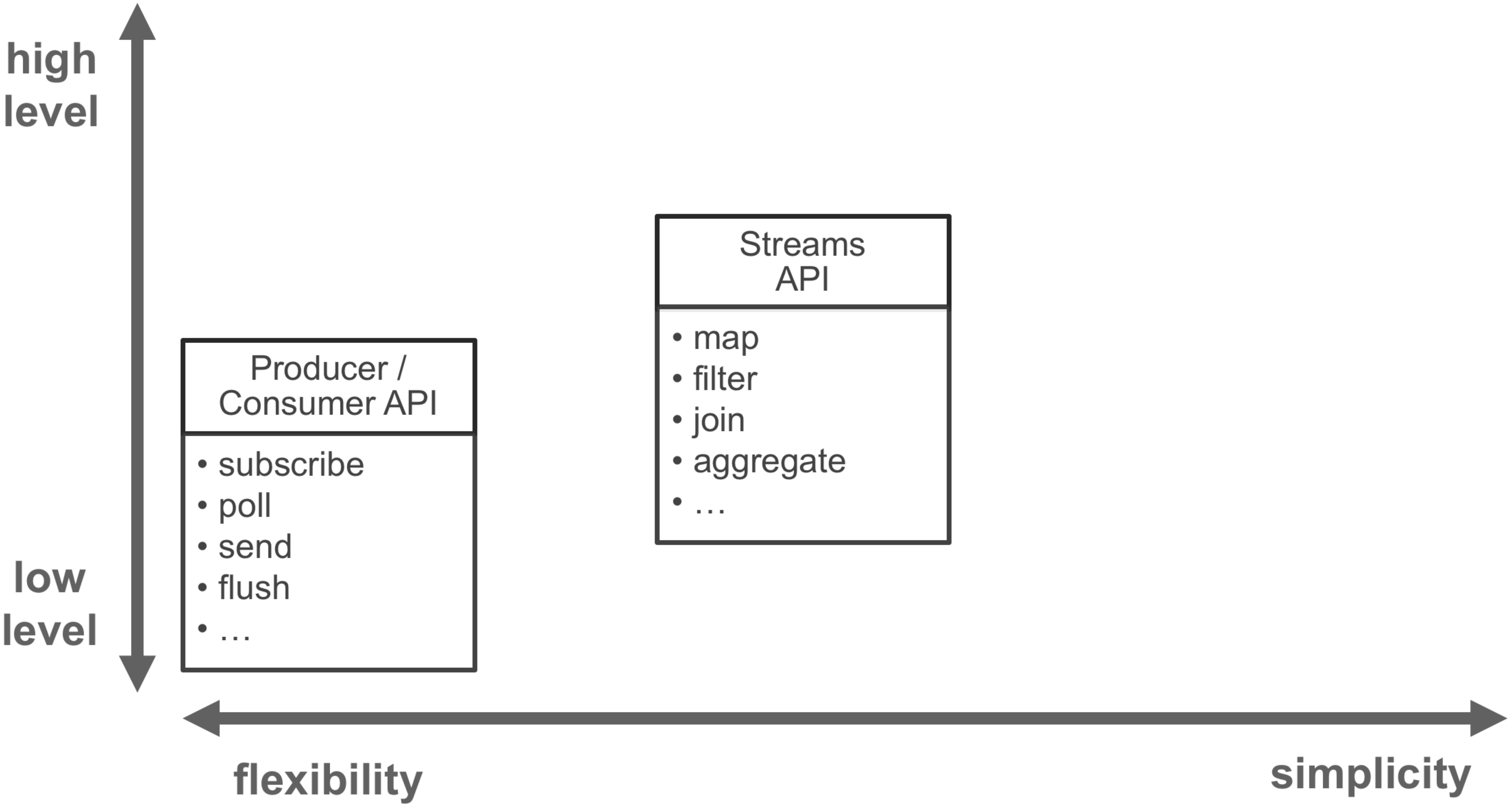
high level

low level

flexibility

simplicity


high level
low level
Producer / Consumer API
• subscribe
• poll
• send
• flush
• …
flexibility
simplicity



high level

low level

Producer / Consumer API

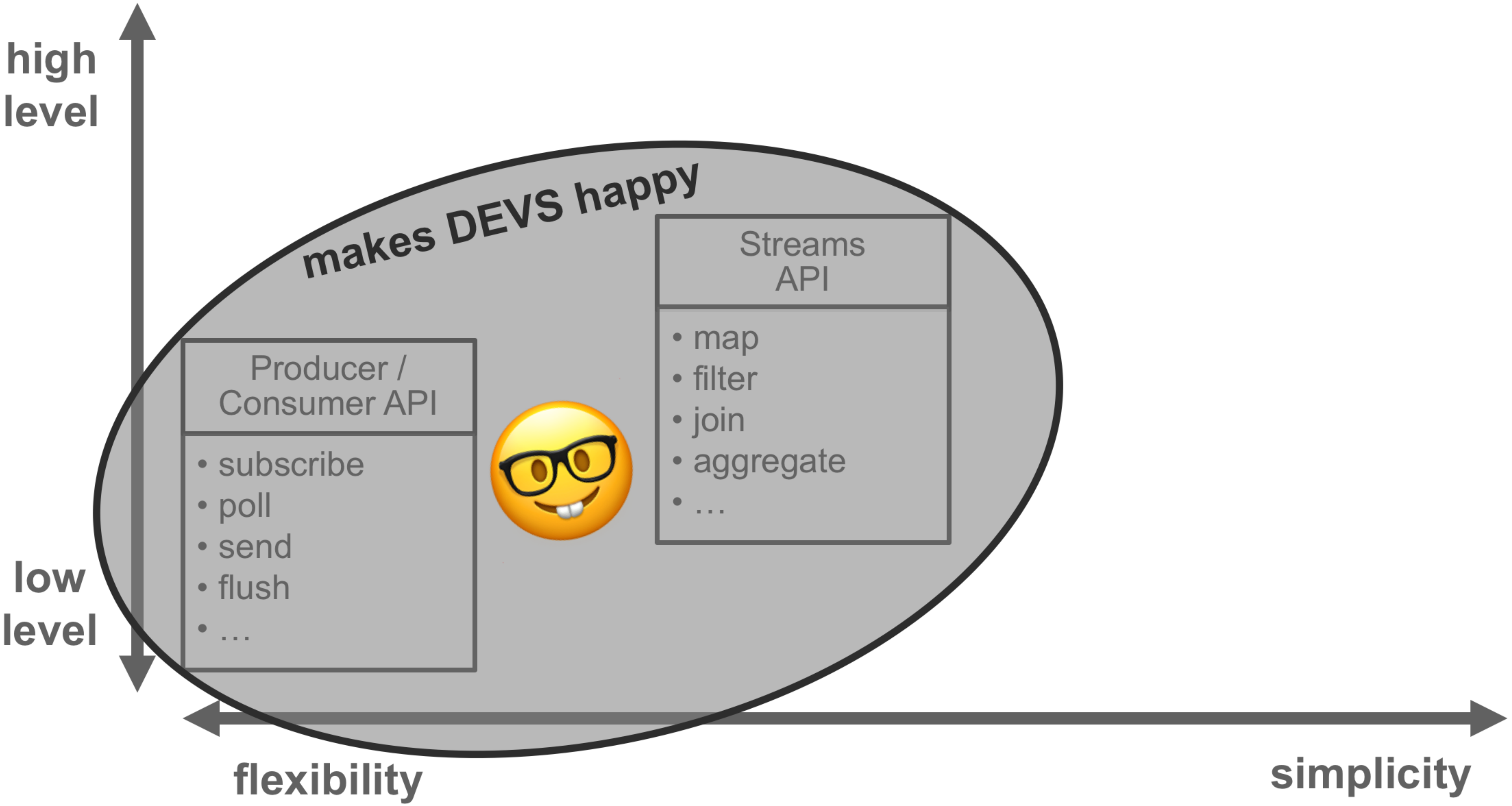
- subscribe
- poll
- send
- flush
- …

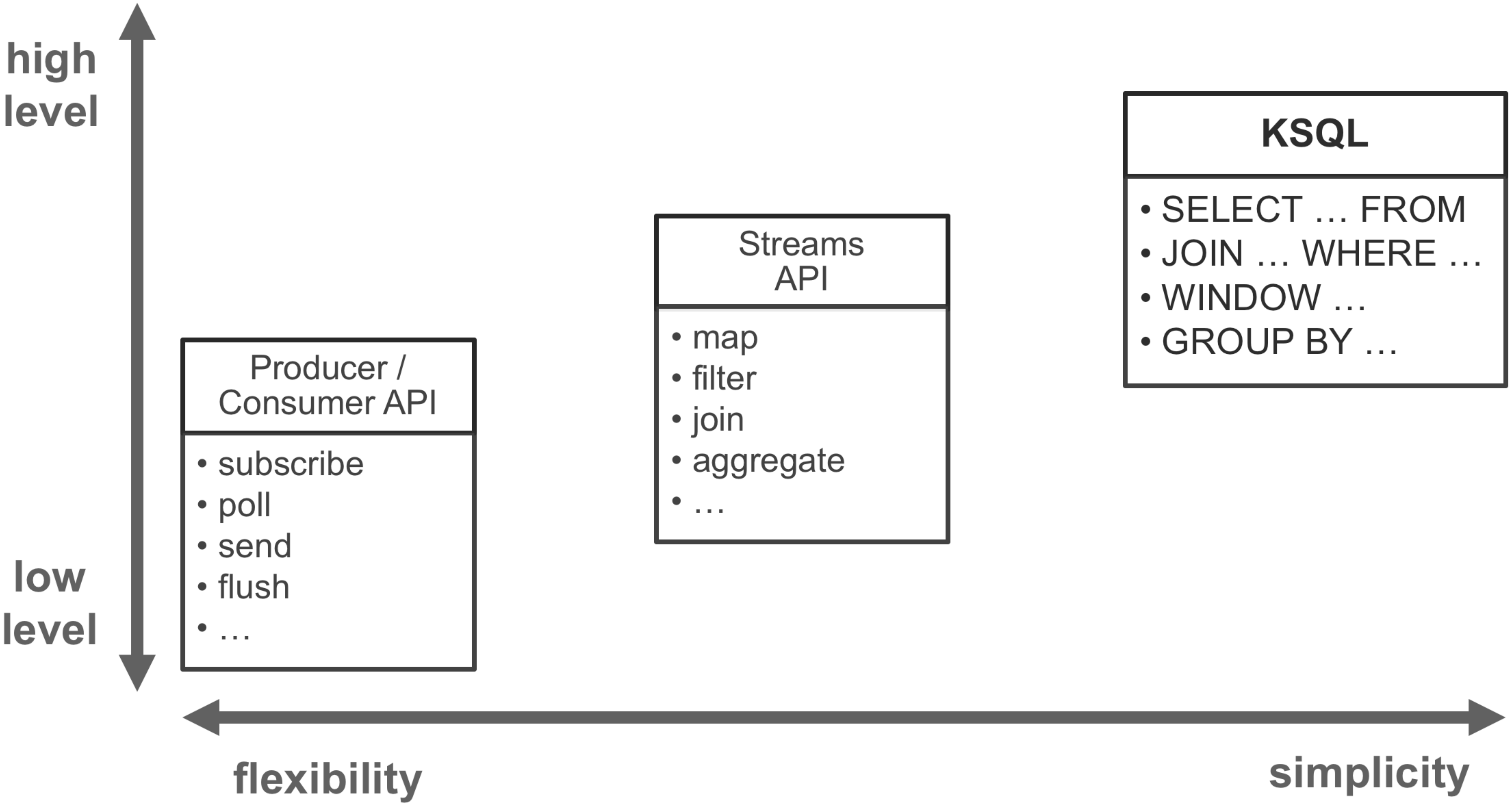
Streams API

- map
- filter
- join
- aggregate
- …

flexibility

simplicity


high level
low level
flexibility
simplicity
makes DEVS happy
Producer / Consumer API
• subscribe
• poll
• send
• flush
• …
Streams API
• map
• filter
• join
• aggregate
• …



high level

low level

flexibility

simplicity

Producer / Consumer API

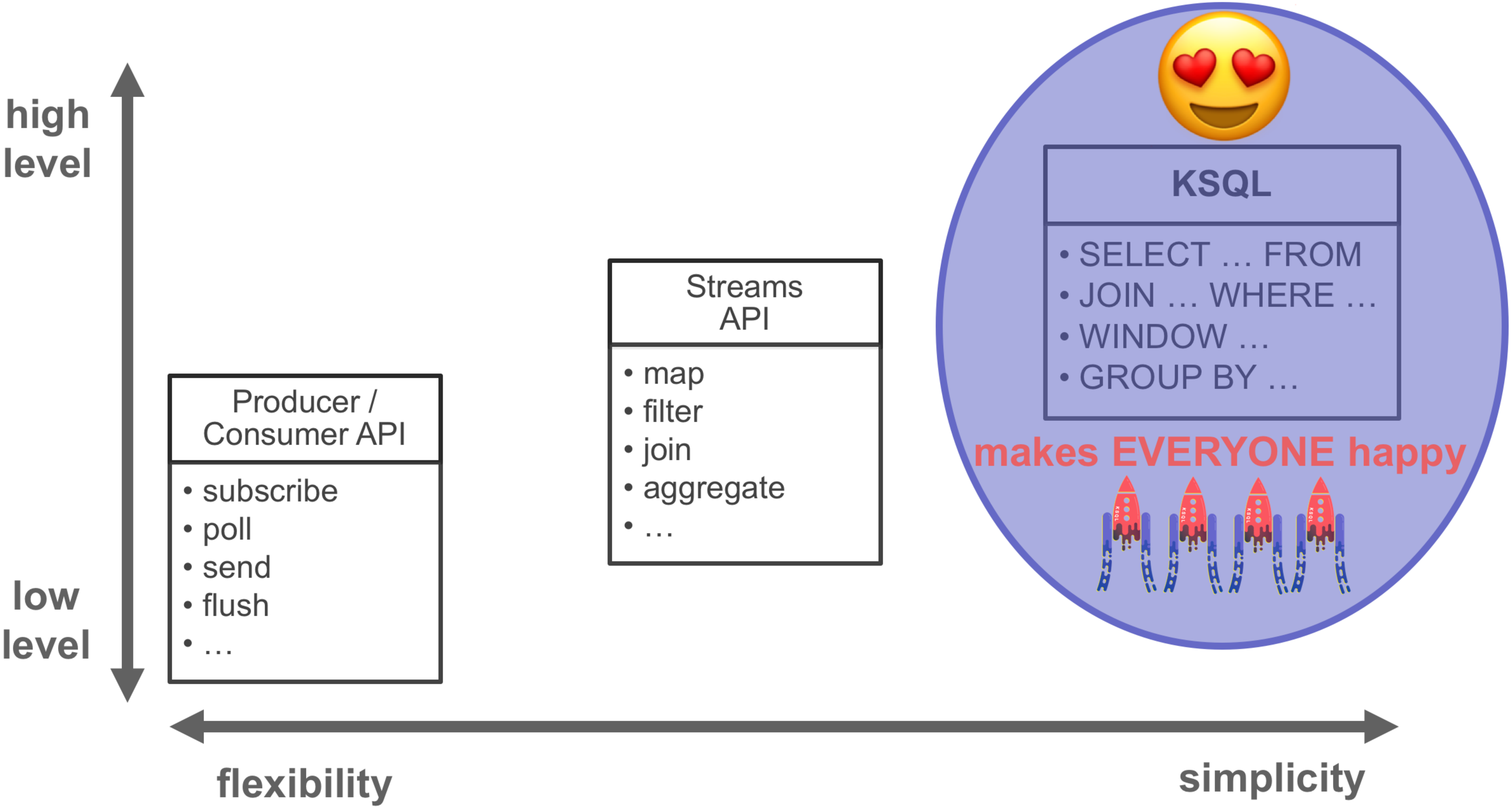
- subscribe
- poll
- send
- flush
- …

Streams API

- map
- filter
- join
- aggregate
- …

KSQL

- SELECT … FROM
- JOIN … WHERE …
- WINDOW …
- GROUP BY …



high level

low level

Producer / Consumer API

- subscribe
- poll
- send
- flush
- …

Streams API

- map
- filter
- join
- aggregate
- …

KSQL

- SELECT … FROM
- JOIN … WHERE …
- WINDOW …
- GROUP BY …

makes EVERYONE happy

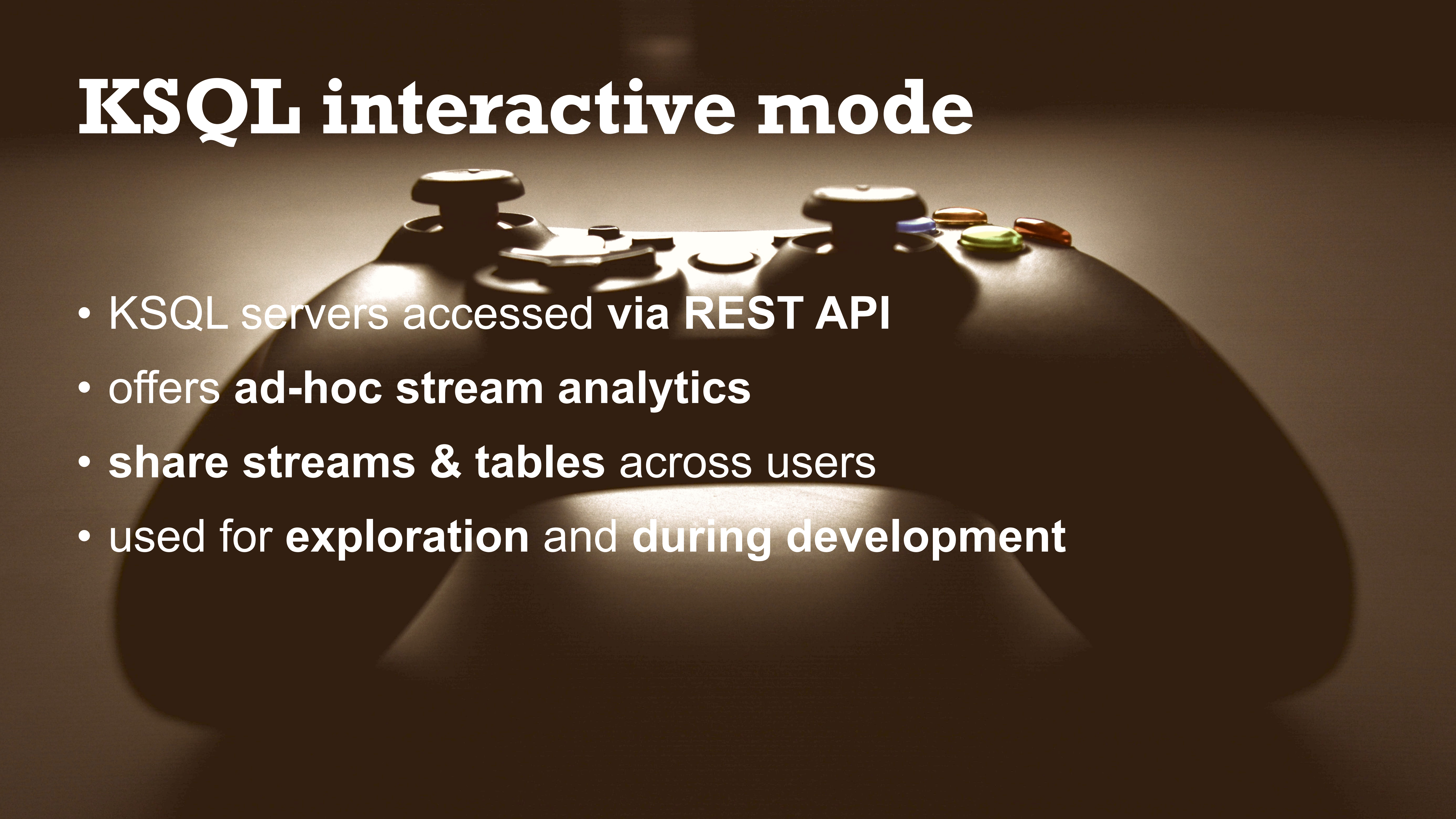
flexibility

simplicity

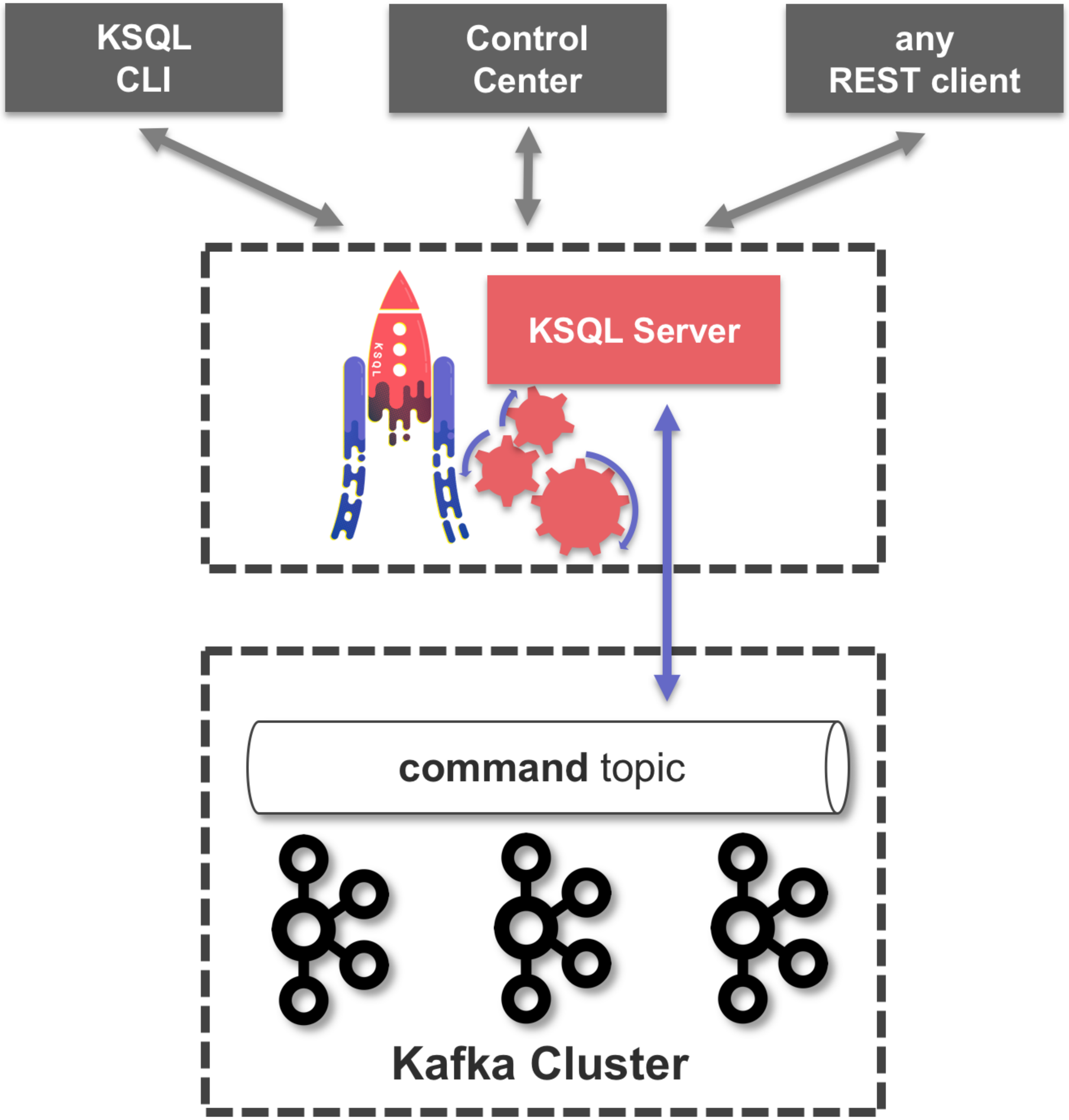
KSQL Queries

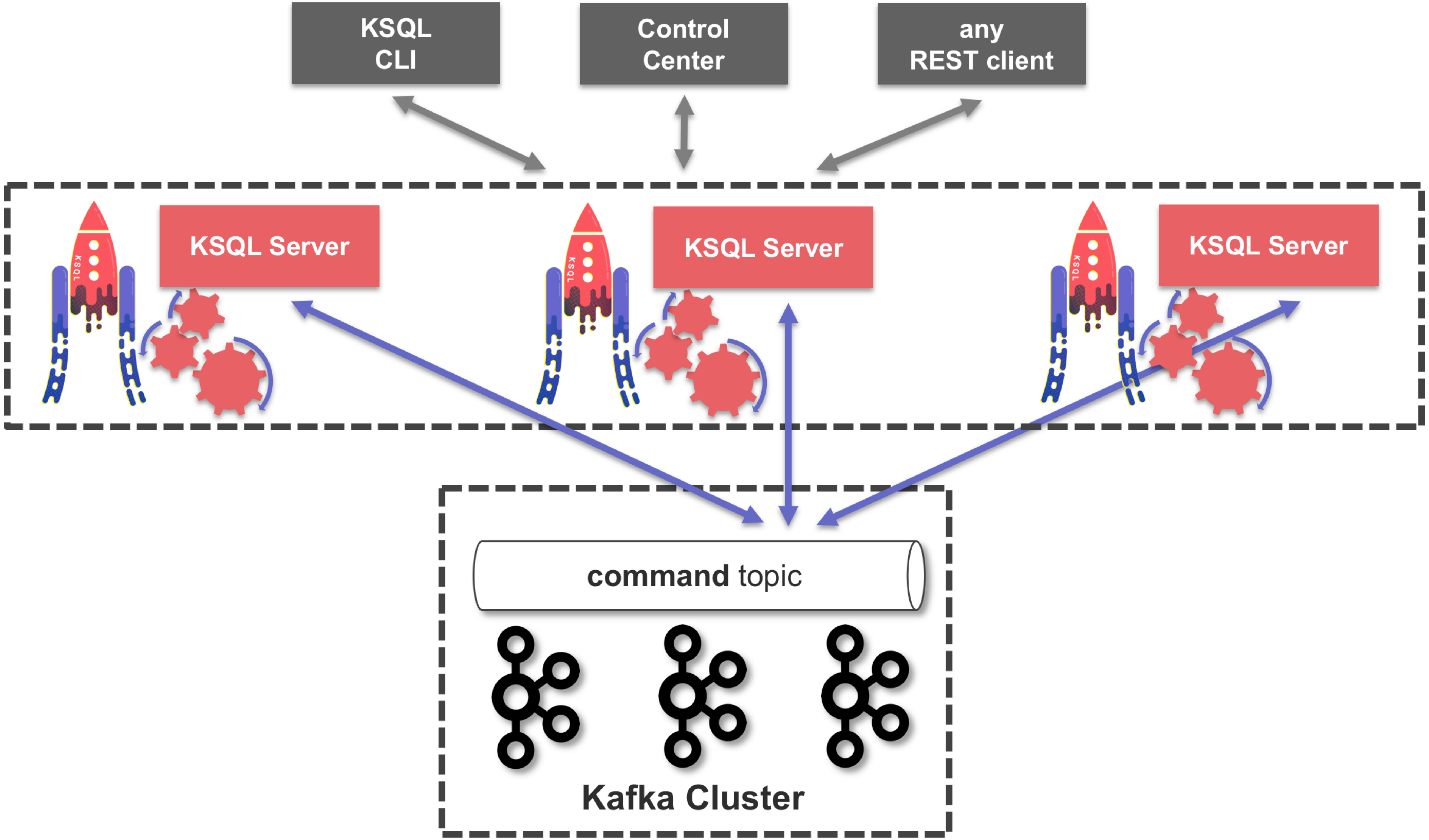
- per-record streaming with **milliseconds latency**
- compiled into **Kafka Streams** applications
- follow same **execution model**
- **distributed** over multiple KSQL servers
- two operation modes / deployment options:
    - **interactive** vs. **headless**

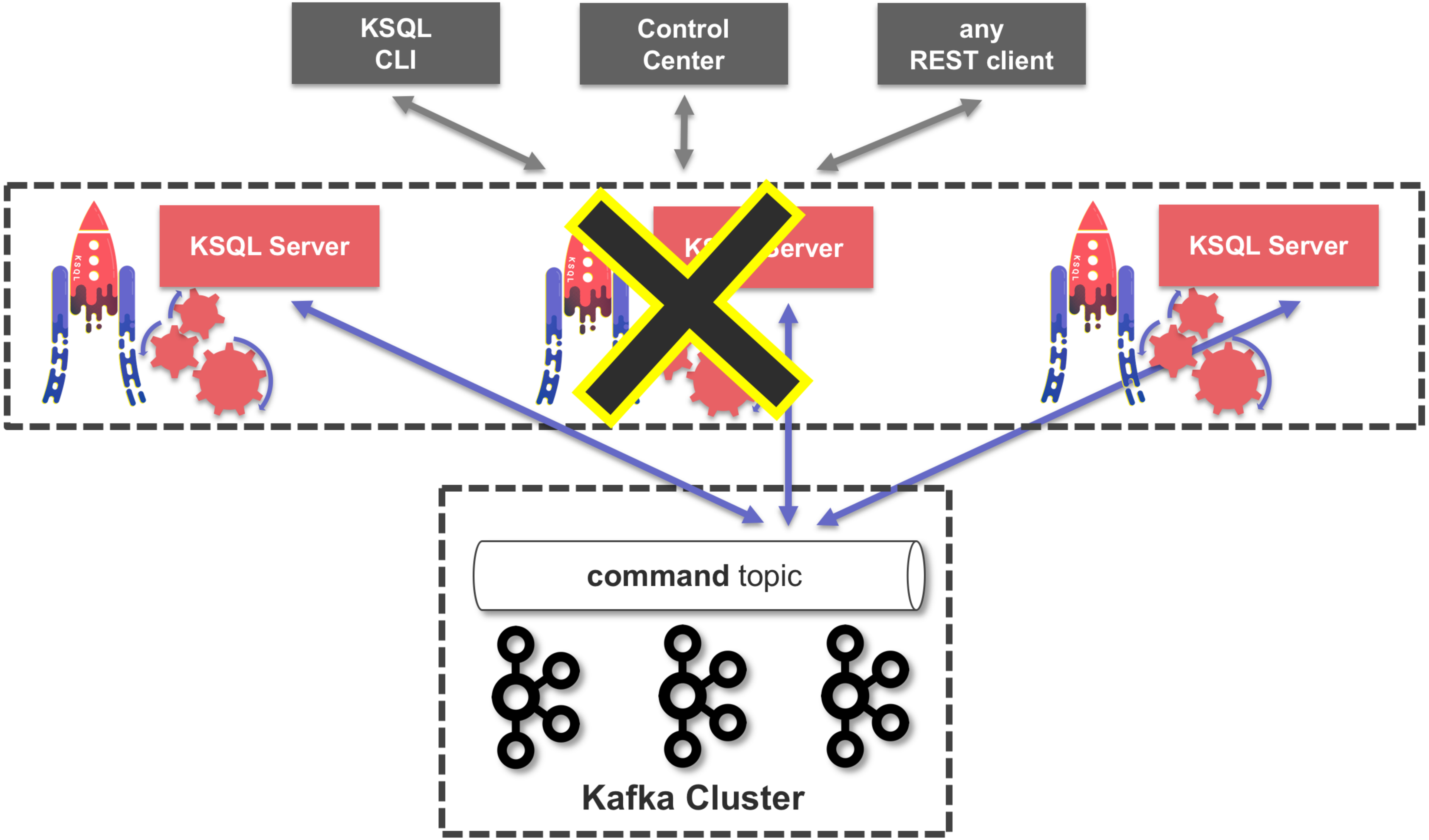
KSQL interactive mode

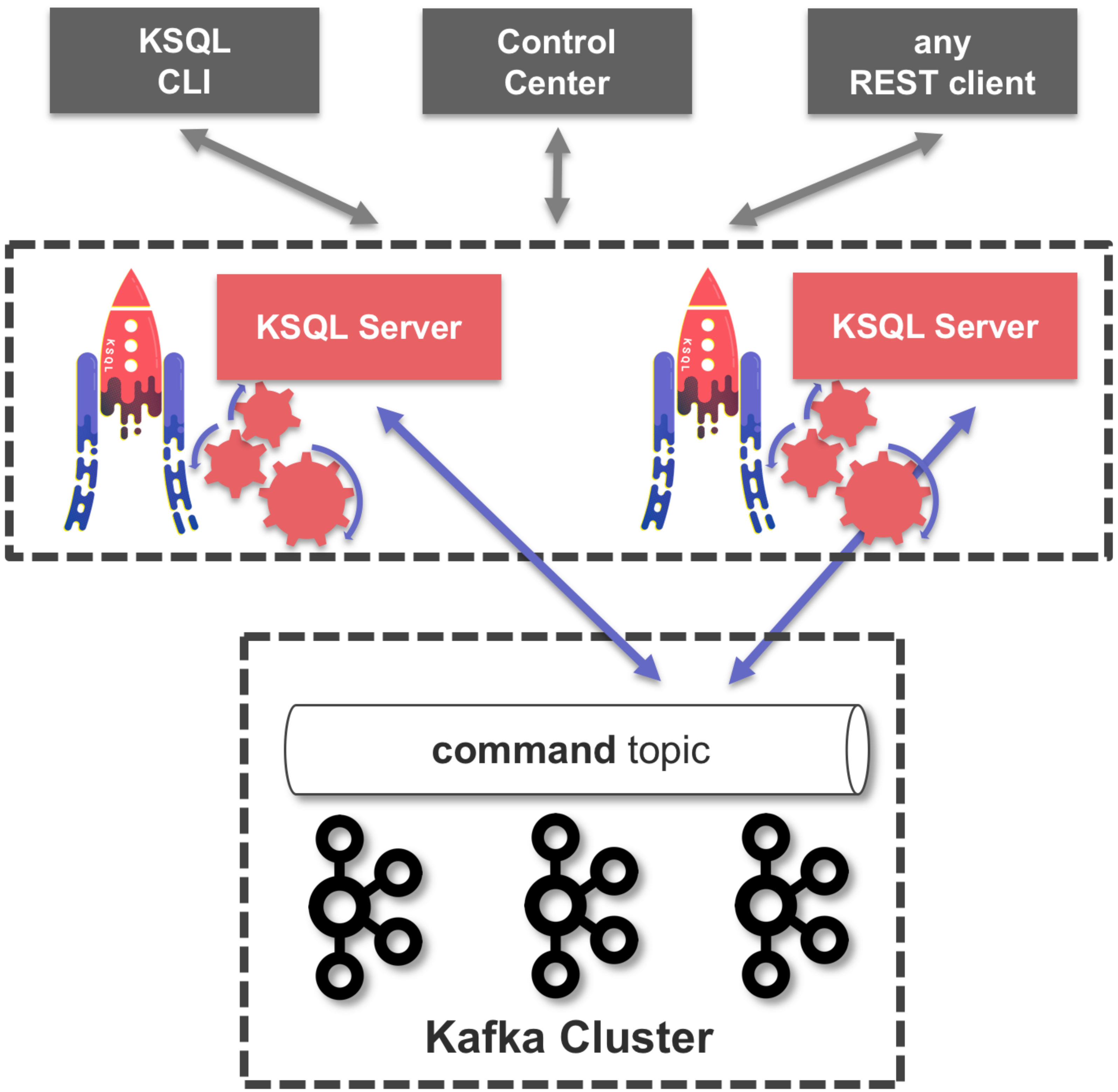


- KSQL servers accessed **via REST API**
- offers **ad-hoc stream analytics**
- **share streams & tables** across users
- used for **exploration** and **during development**

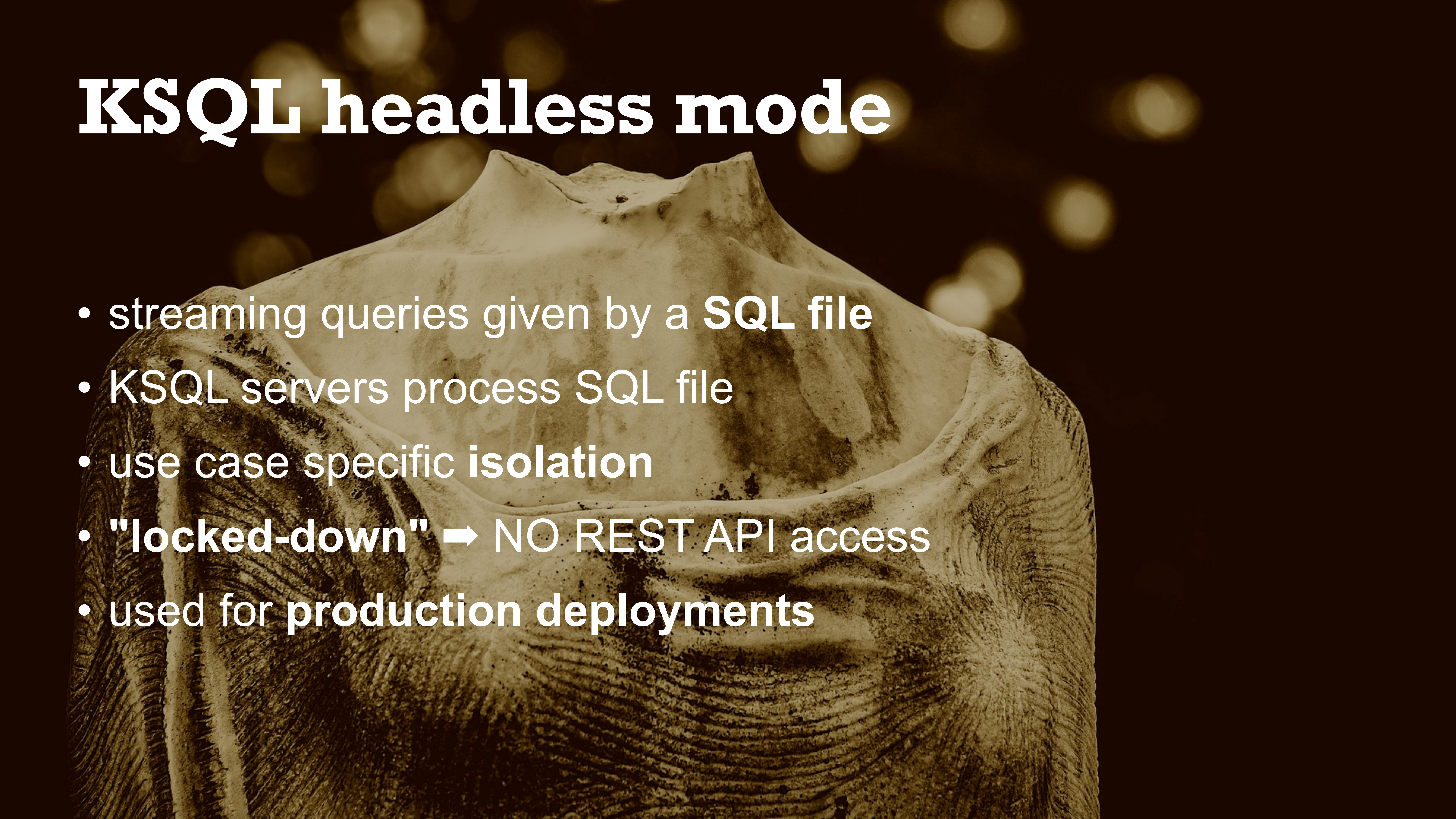

KSQL CLI
Control Center
any REST client
KSQL Server
KSQL
command topic
Kafka Cluster


KSQL
CLI
Control
Center
any
REST client
KSQL Server
KSQL Server
KSQL Server
command topic
Kafka Cluster

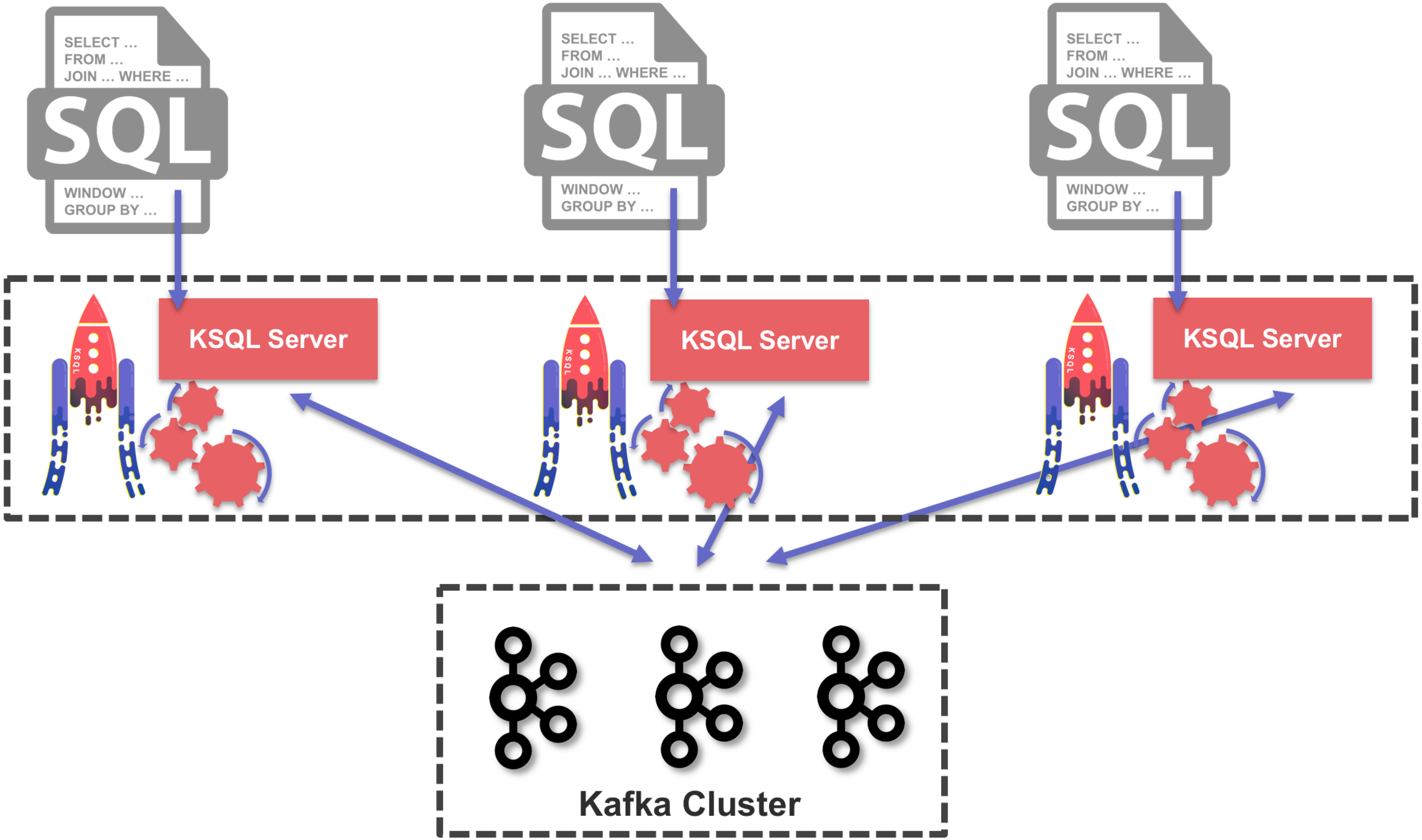

KSQL CLI
Control Center
any REST client
KSQL Server
KS Server
KSQL Server
KSQL
command topic
Kafka Cluster

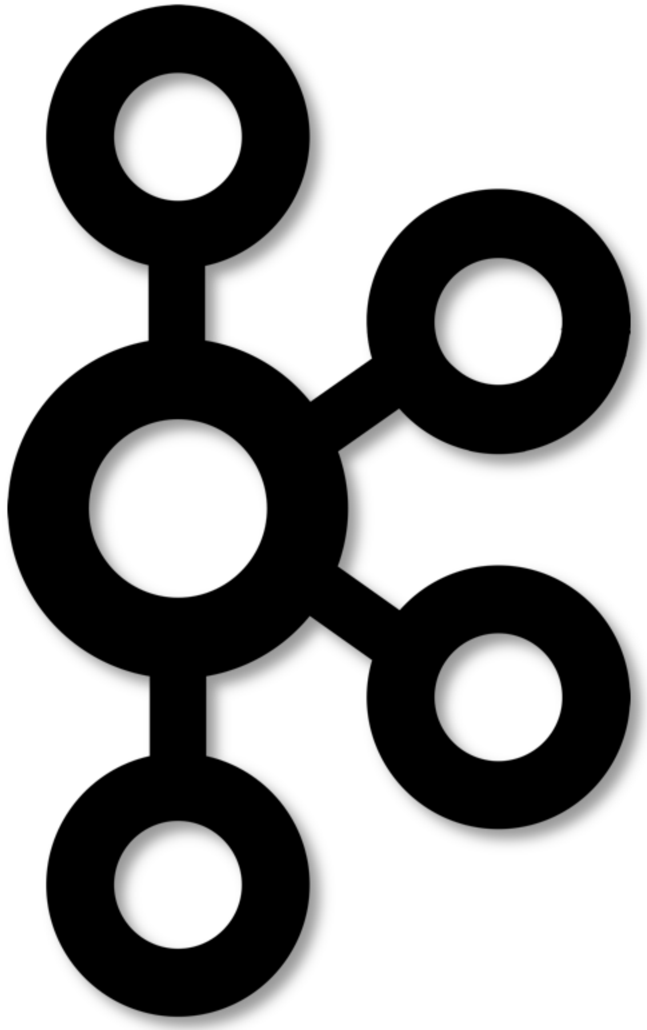

KSQL CLI
Control Center
any REST client
KSQL Server
KSQL Server
command topic
Kafka Cluster

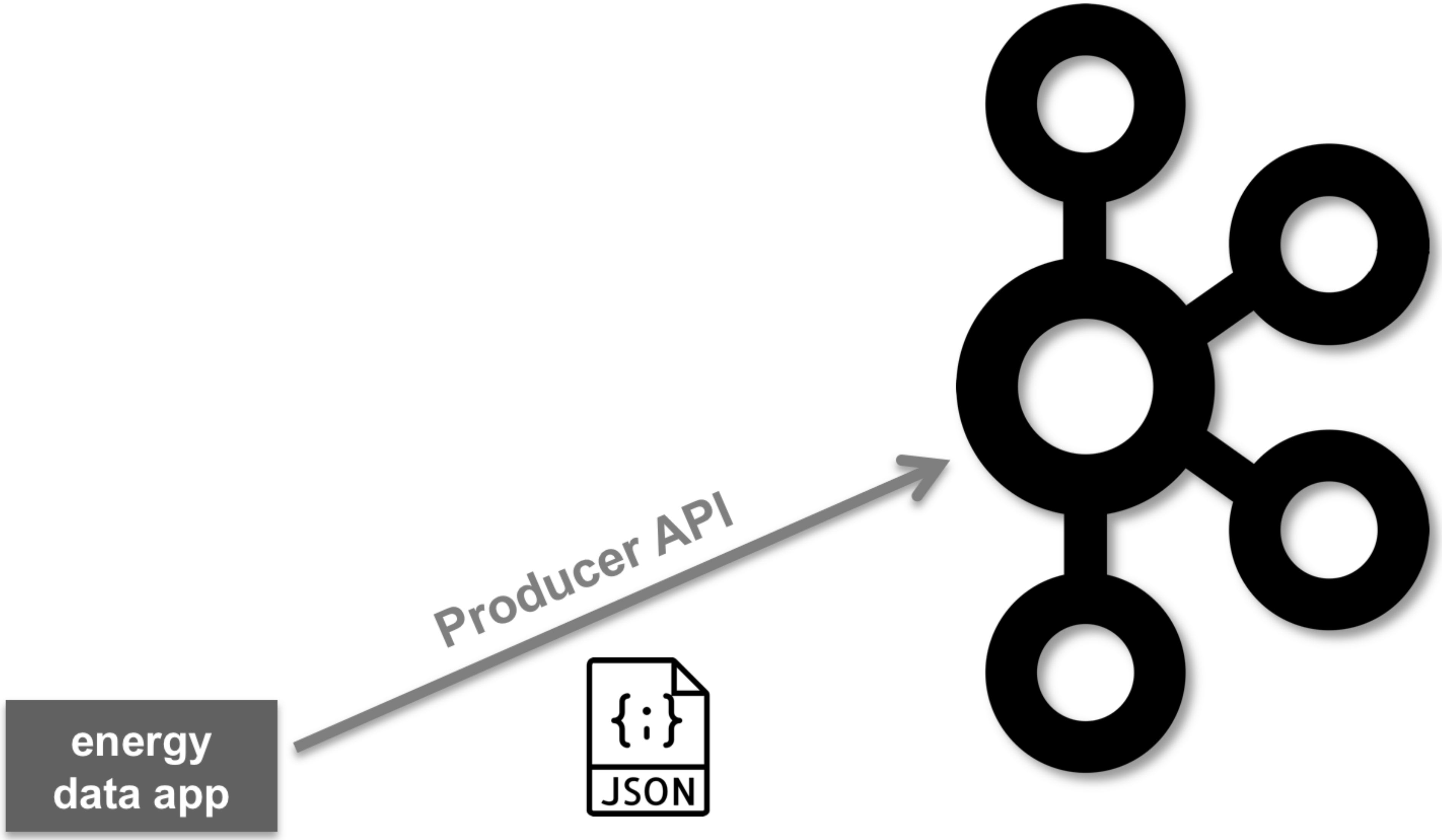
KSQL headless mode

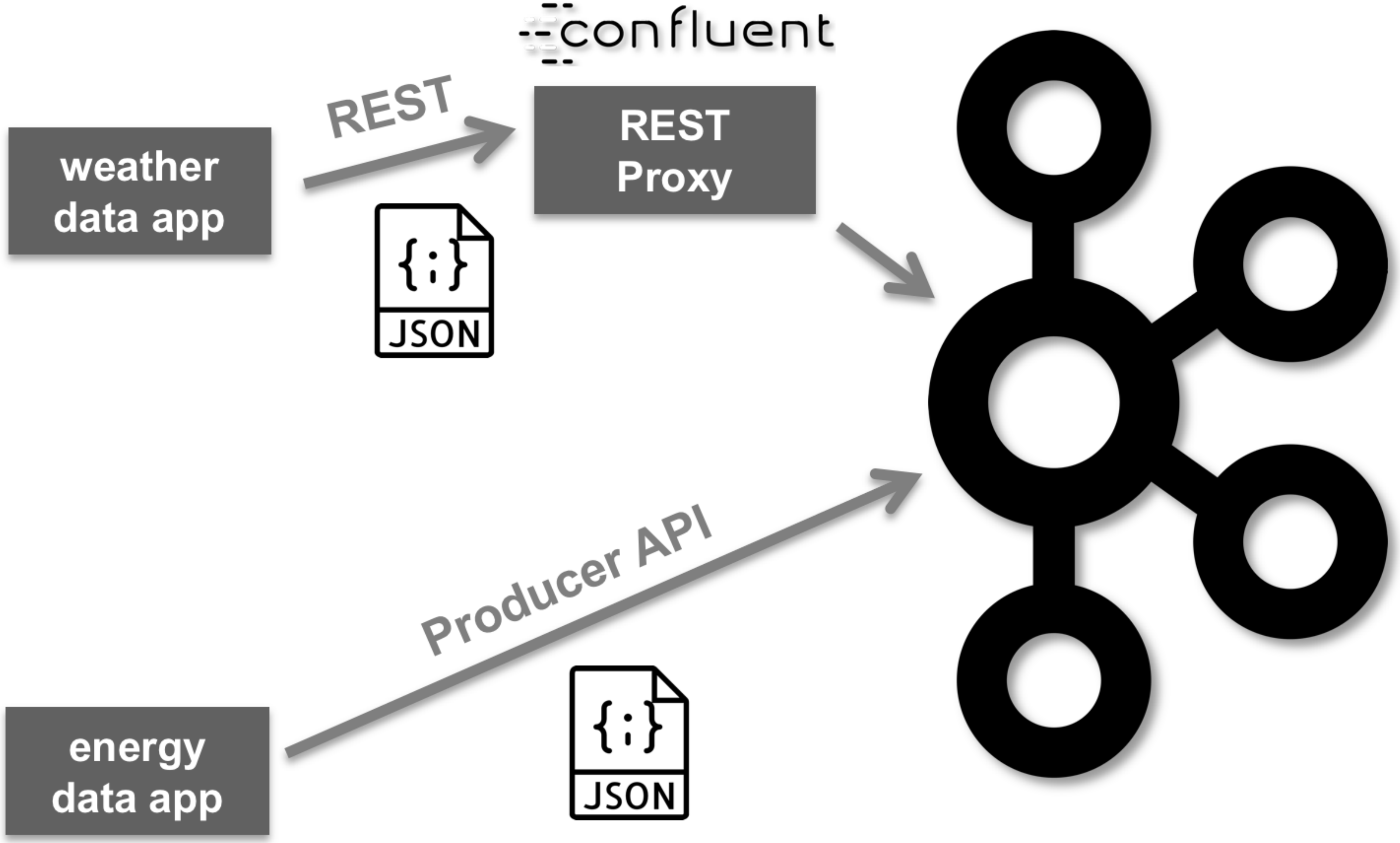


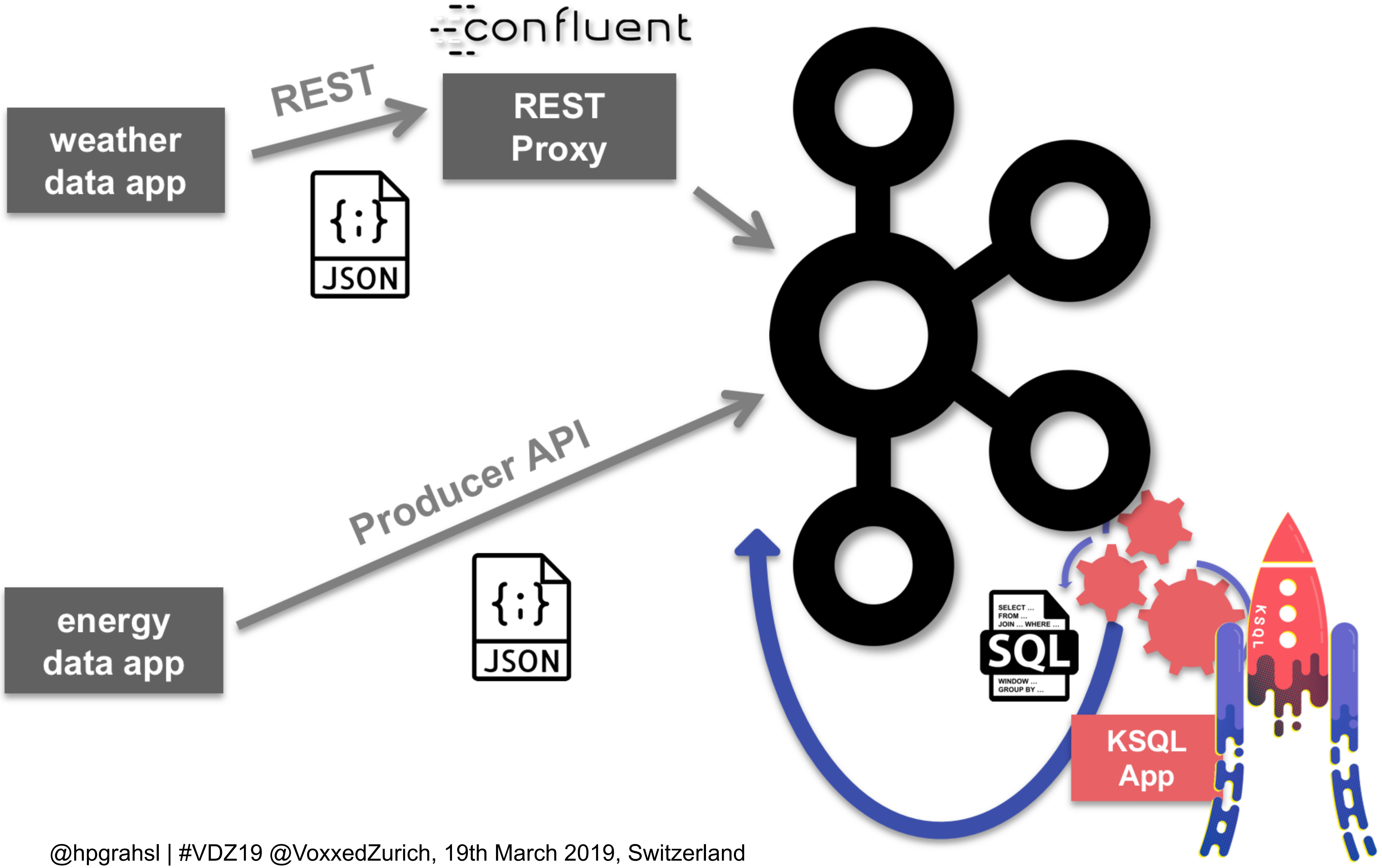
- streaming queries given by a **SQL file**
- KSQL servers process SQL file
- use case specific **isolation**
- **"locked-down"** ➡ NO REST API access
- used for **production deployments**

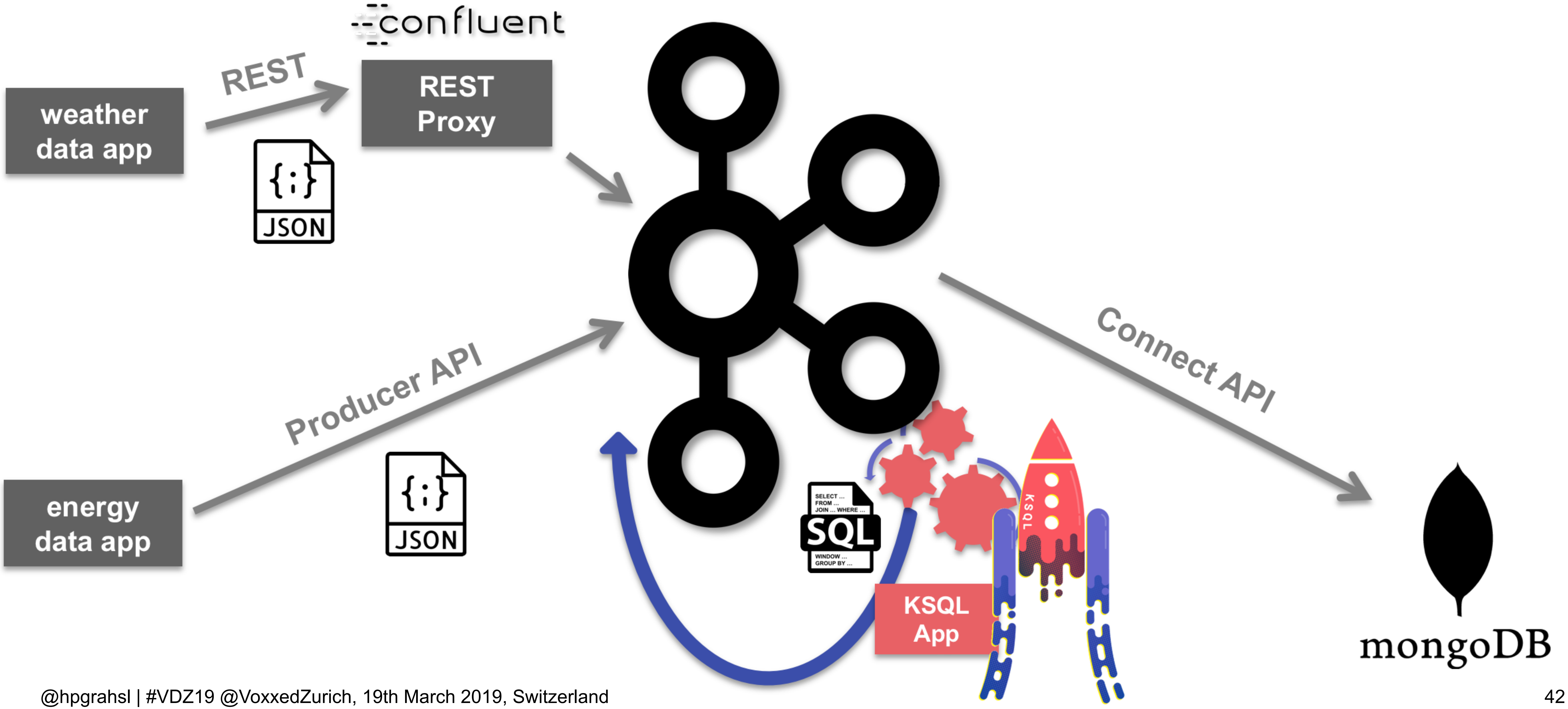

SELECT …
FROM …
JOIN … WHERE …
SQL
WINDOW …
GROUP BY …
SELECT …
FROM …
JOIN … WHERE …
SQL
WINDOW …
GROUP BY …
SELECT …
FROM …
JOIN … WHERE …
SQL
WINDOW …
GROUP BY …
KSQL Server
KSQL Server
KSQL Server
Kafka Cluster

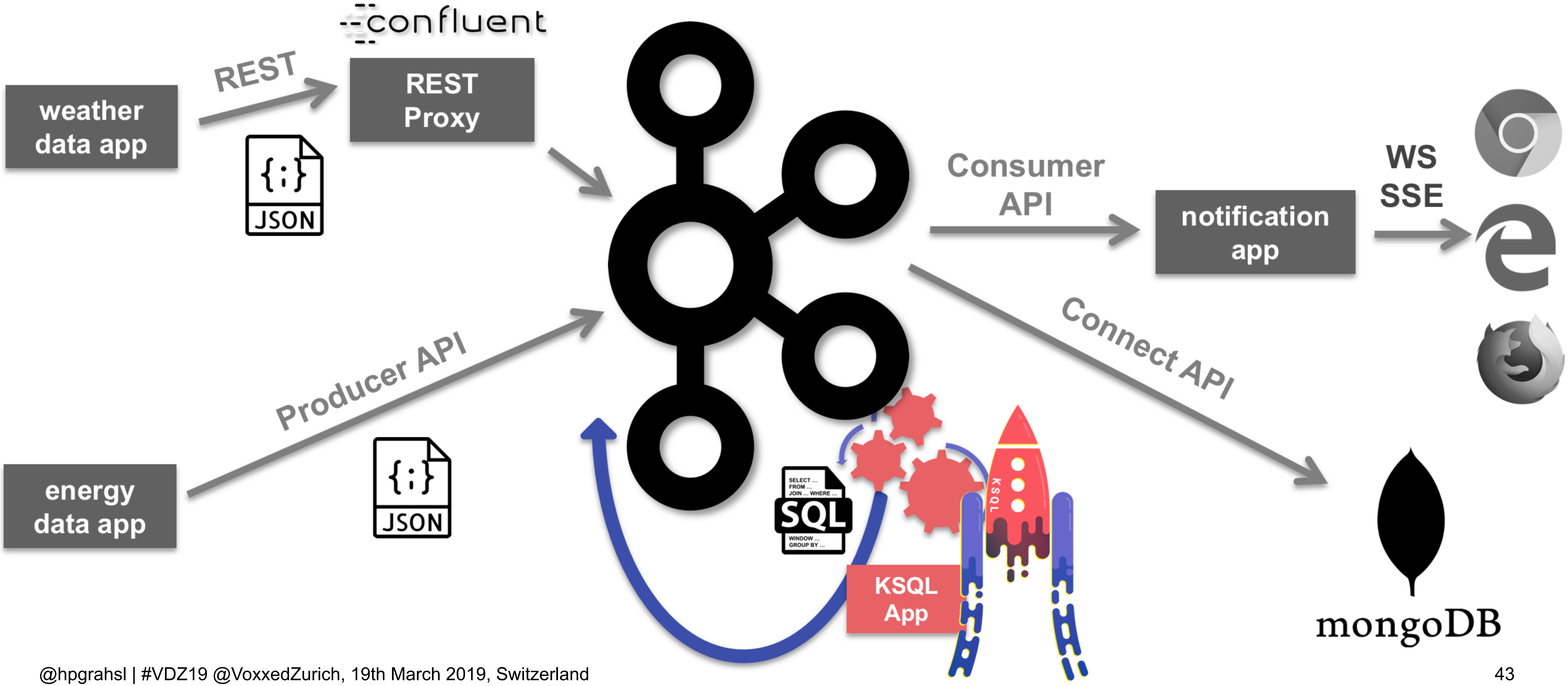



energy data app
Producer API
JSON


confluent
weather data app
REST
REST Proxy
JSON
Producer API
energy data app
JSON


confluent
weather data app
REST
REST Proxy
JSON
energy data app
Producer API
JSON
SELECT ... FROM ... JOIN ... WHERE ...
SQL
WINDOW ... GROUP BY ...
KSQL
KSQL App


confluent
weather data app
REST
JSON
REST Proxy
energy data app
Producer API
JSON
SELECT ... FROM ... JOIN ... WHERE ...
SQL
WINDOW ... GROUP BY ...
KSQL App
KSQL
Connect API
mongoDB



confluent

weather data app

REST

REST Proxy

energy data app

Producer API

Consumer API

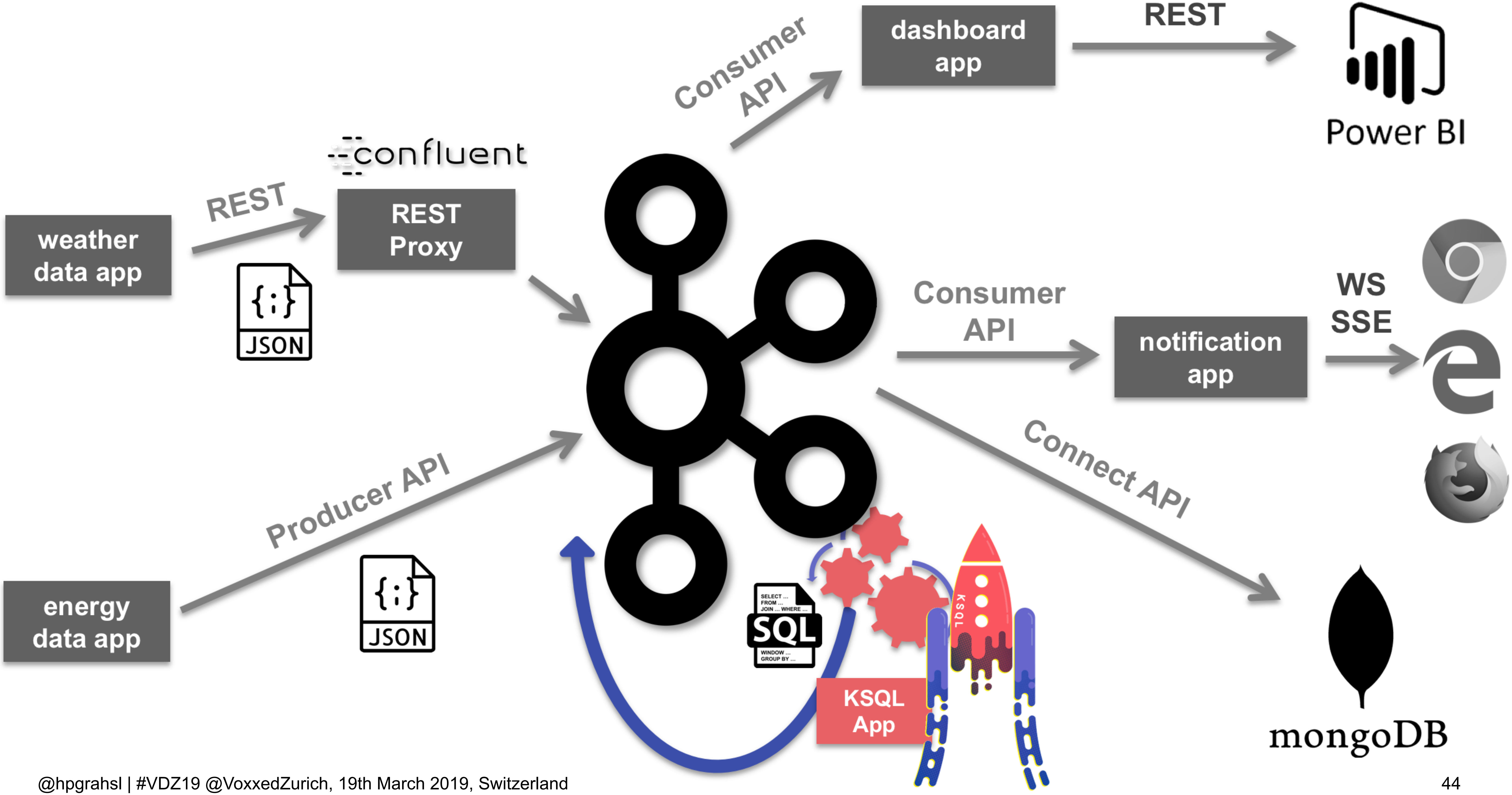
notification app

WS SSE

Connect API

KSQL App

mongoDB

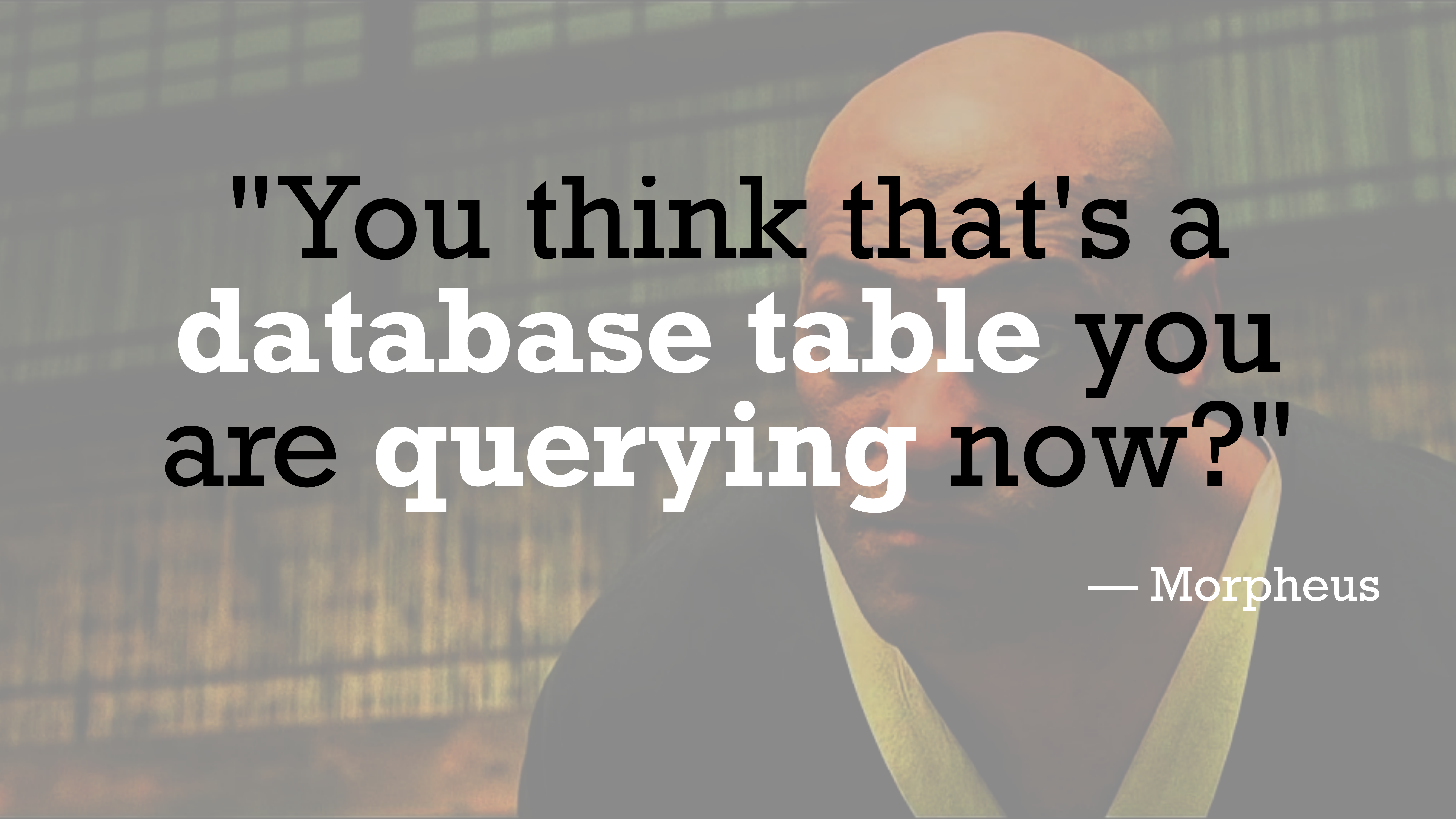

weather data app
REST
confluent
REST Proxy
JSON
energy data app
Producer API
JSON
Consumer API
dashboard app
REST
Power BI
Consumer API
notification app
WS
SSE
Connect API
mongoDB
SELECT ... FROM ... JOIN ... WHERE ...
SQL
WINDOW ... GROUP BY ...
KSQL
KSQL App

step 1

ingest sensor data

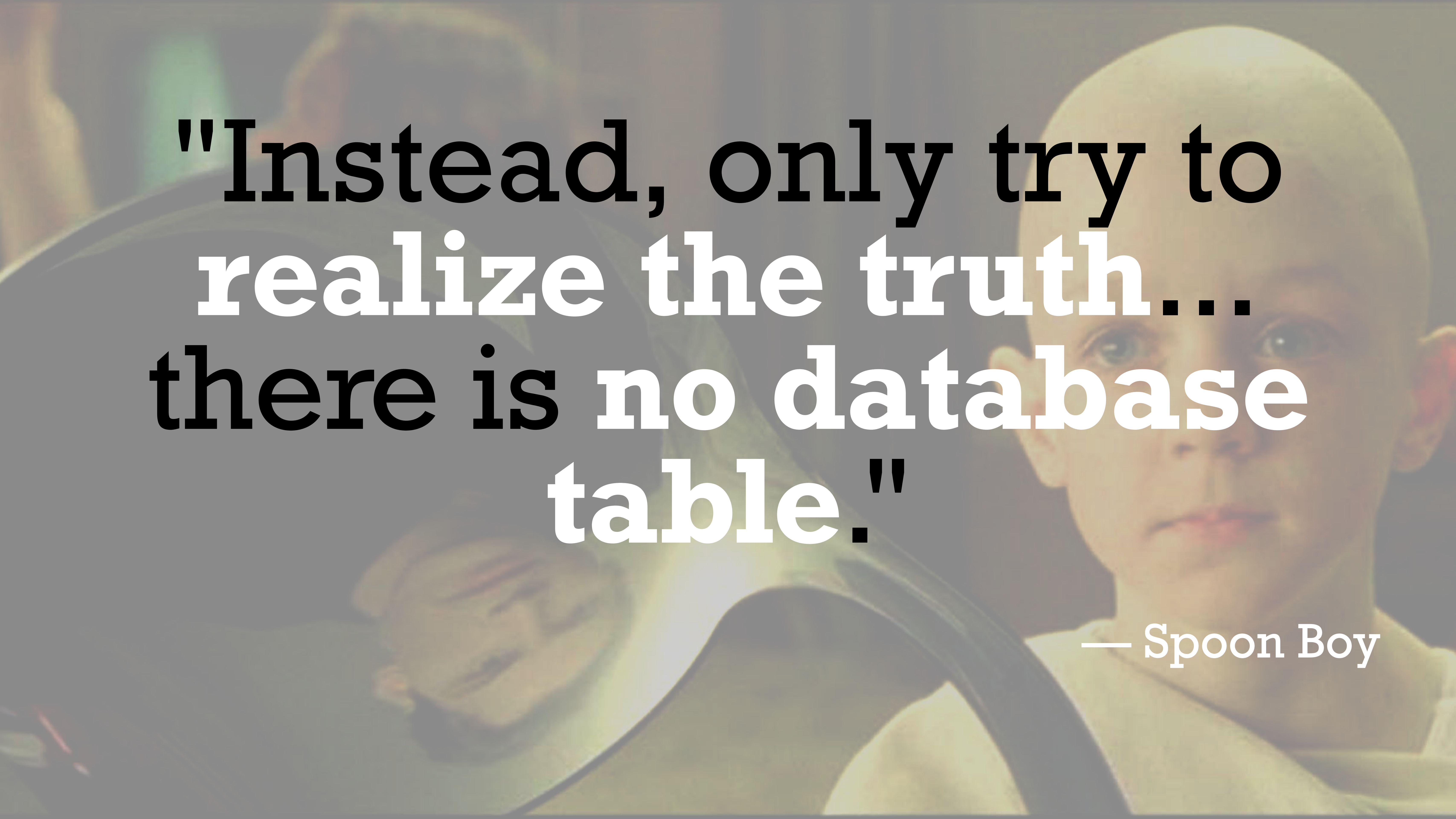
step 2

KSQL streaming



"You think that's a **database table** you are **querying** now?"

— Morpheus



"Instead, only try to **realize the truth**… there is **no database table**."

— Spoon Boy

step 3

connecting NoSQL

step 4

reactive notifications

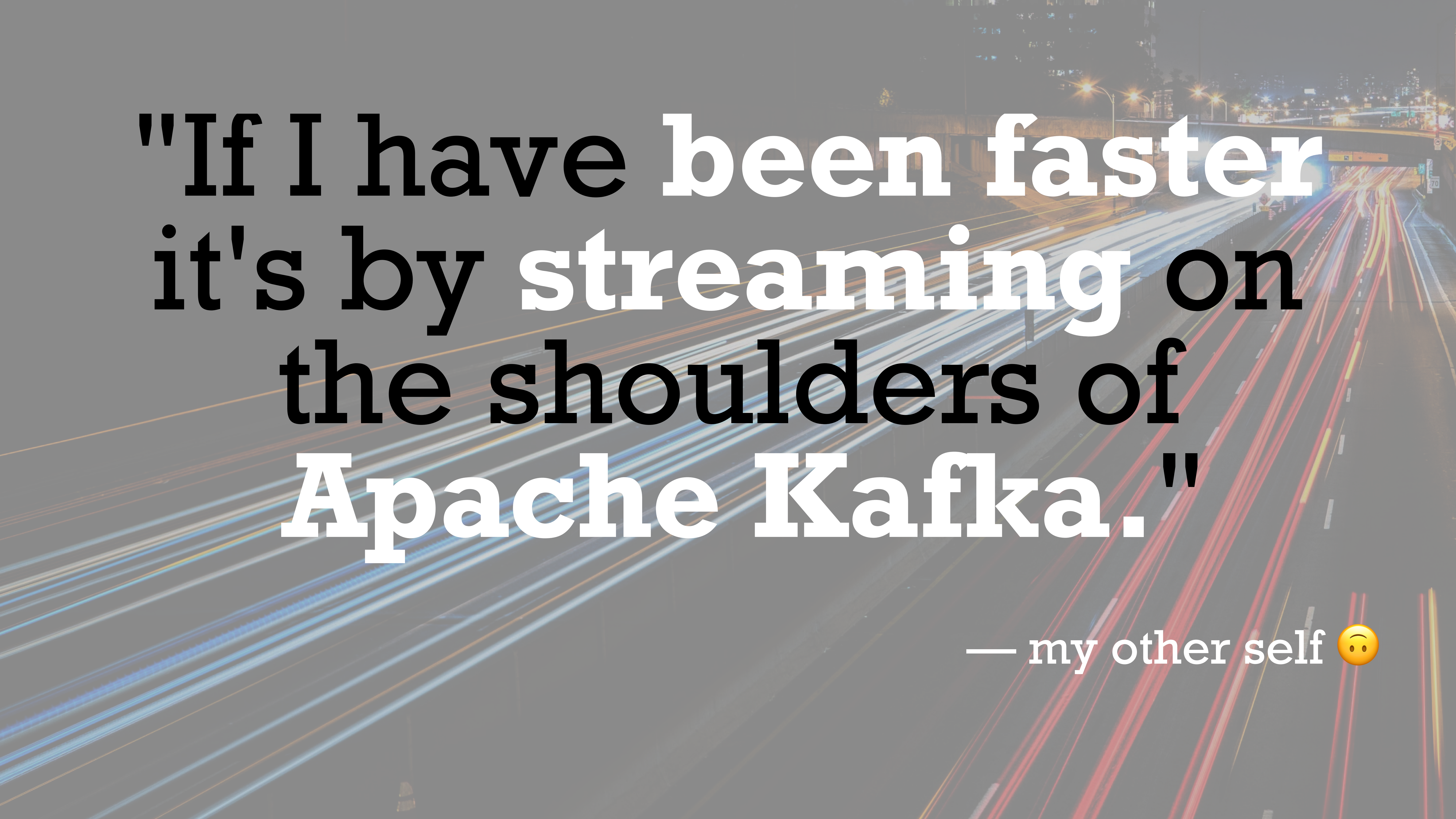
step 5

live dashboards

MISSION accomplished

KSQL wrap-up

- **streaming with SQL**
  ... and nothing but SQL
- **scalable & fault-tolerant**
- deployable **anywhere**: cloud or on prem
- viable for **use cases of any size** (XS ... XXXL)
- **exactly-once** delivery guarantee semantics



"If I have **been faster** it's by **streaming** on the shoulders of **Apache Kafka.**"

— my other self 🙃

Your obsession tells you to do batching.
I tell you to walk away and stream with

KSQL

The choice is yours folks!

THANK YOU

Q & A ?

https://bit.ly/2FaLr7w

NETCONOMY