Building event-driven applications with Confluent Platform for Cloud Pak for Integration

Technical Overview

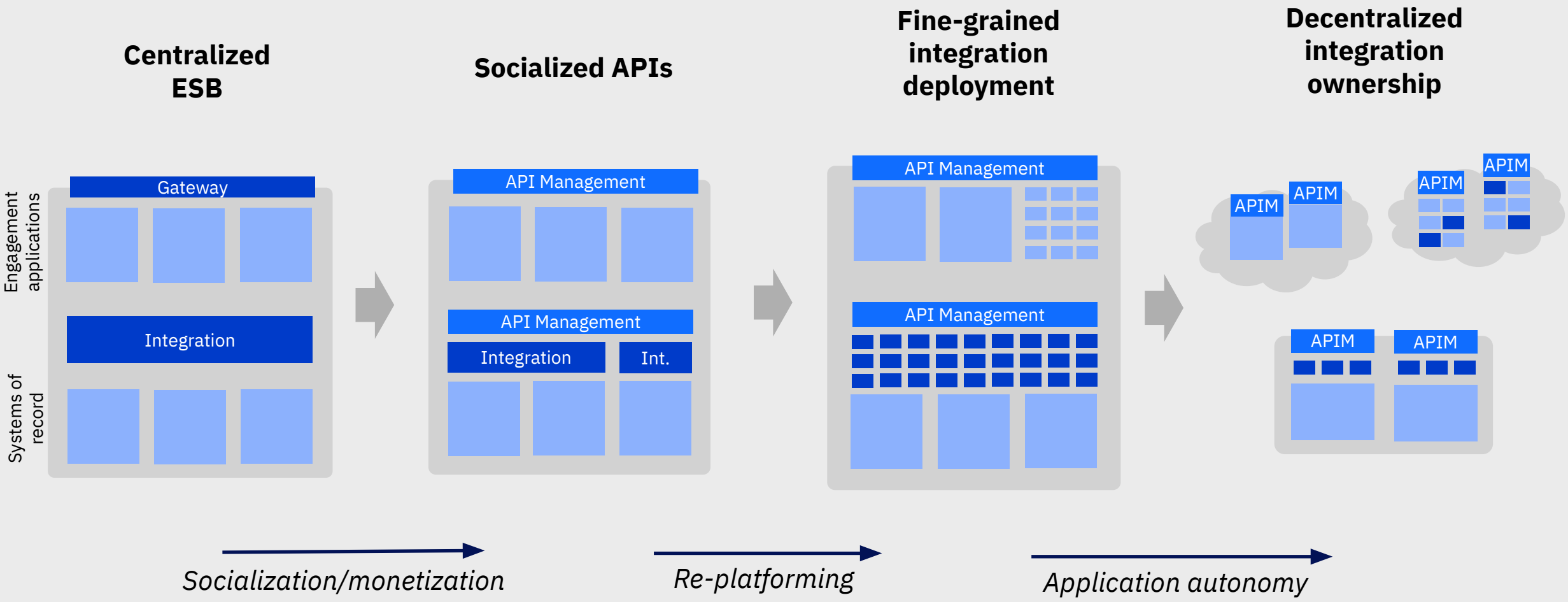
Kim Clark
Integration Architect

IBM Cloud

Jeremy Hogan
Cloud Partner Solution Architect

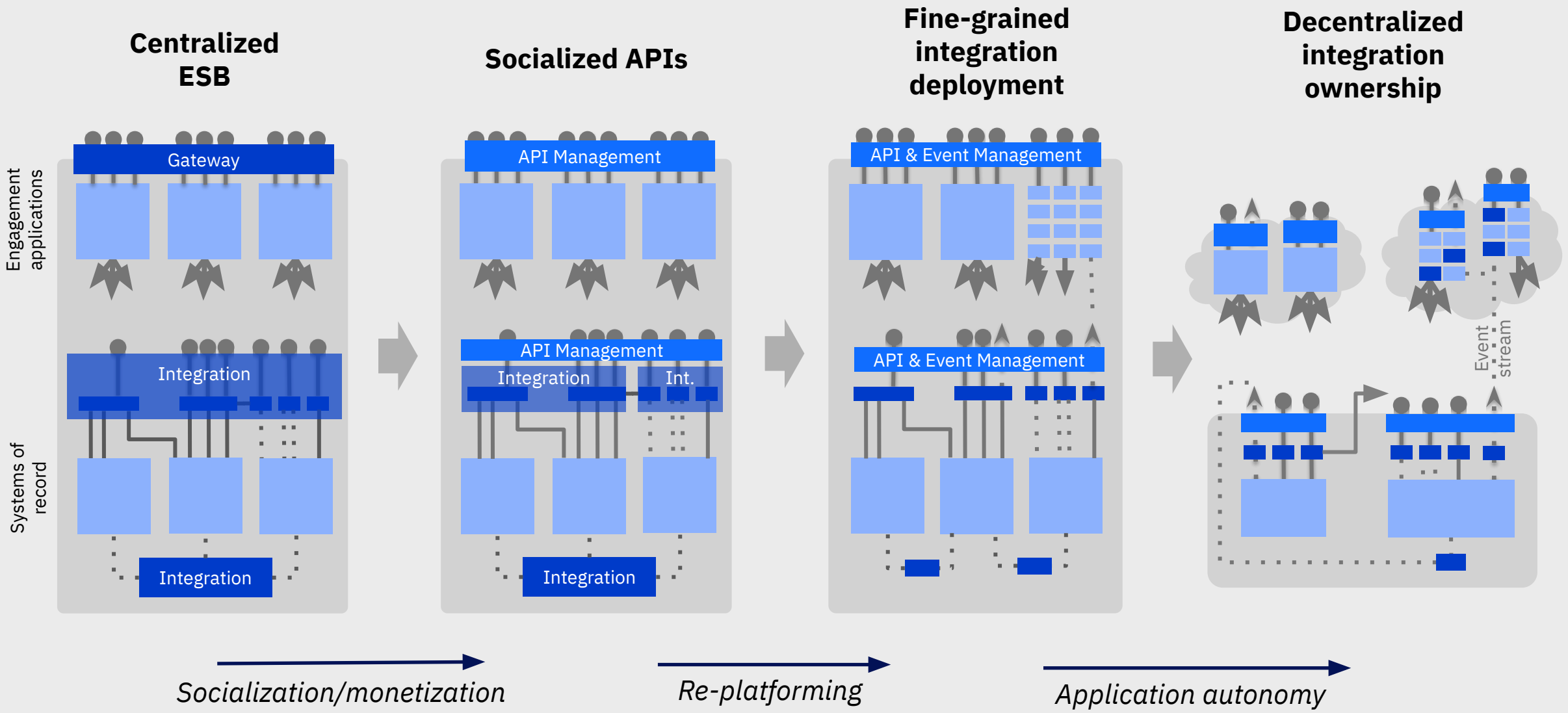
CONFLUENT

Evolution to **agile integration** – high level view



Webinar http://ibm.biz/agile-integration-webinar eBooklet http://ibm.biz/agile-integration IBM Redbook http://ibm.biz/agile-integration-redbook

Evolution to **agile integration** – detail view

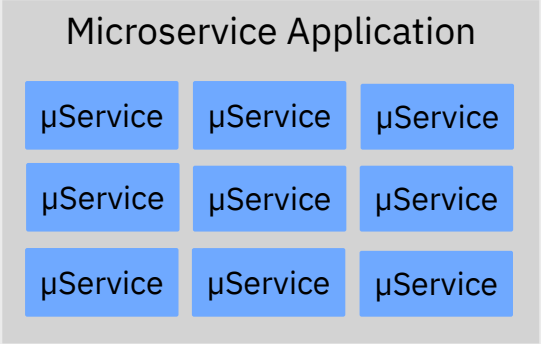


Webinar http://ibm.biz/agile-integration-webinar

eBooklet http://ibm.biz/agile-integration

IBM Redbook http://ibm.biz/agile-integration-redbook

Creating truly independent digital applications requires asynchronous communication as well as APIs



Truly independent, decoupled microservice components enable

Agility

Innovate rapidly without affecting other components

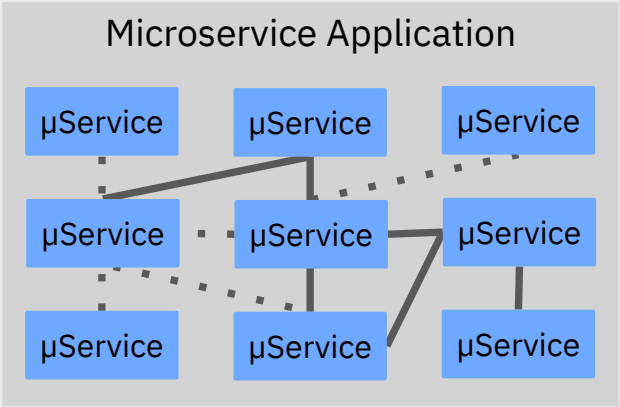
Scalability

Scale only what you need, and only when you need to

Resilience

Fail fast, return fast, without affecting other components

Creating truly independent digital applications requires asynchronous communication as well as APIs



Truly independent, decoupled microservice components enable

Agility

Innovate rapidly without affecting other components

Scalability

Scale only what you need, and only when you need to

Resilience

Fail fast, return fast, without affecting other components

To provide those benefits they need to be independent of one another, and from the systems of record

APIs

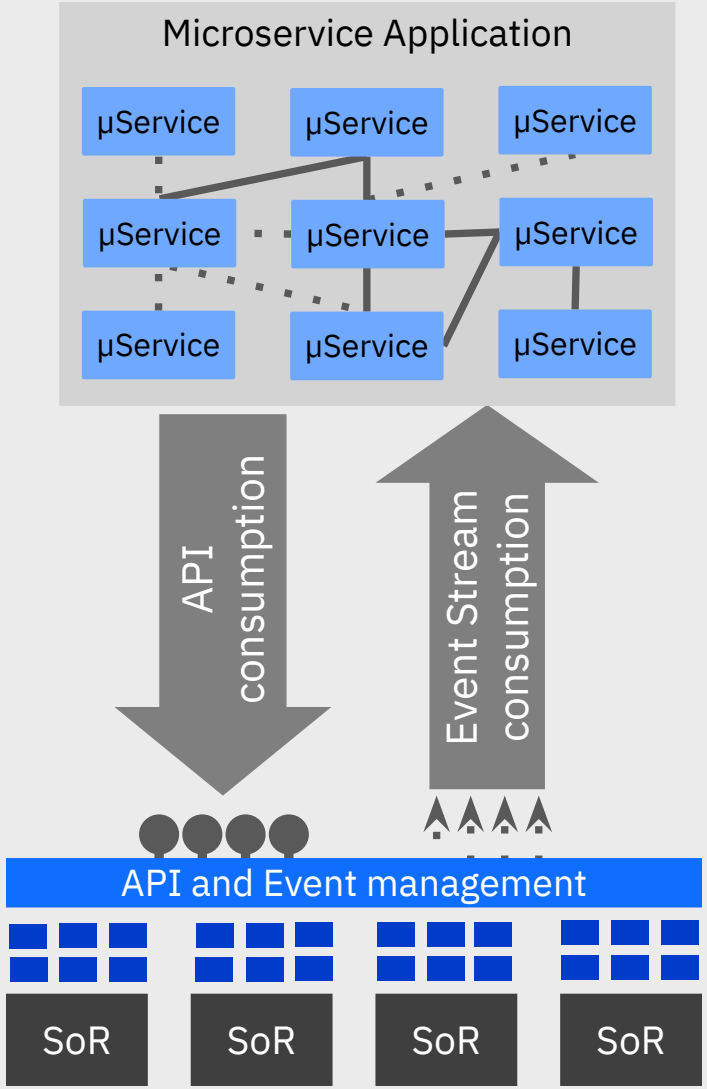
Are simplest to use, but create a real-time dependency on the underlying system of record

Event streams

Enable microservices to build decoupled views of the data and respond to real time events

http://ibm.biz/eda-resurgence-slides

Creating truly independent digital applications requires asynchronous communication as well as APIs



Truly independent, decoupled microservice components enable

Agility

Innovate rapidly without affecting other components

Scalability

Scale only what you need, and only when you need to

Resilience

Fail fast, return fast, without affecting other components

To provide those benefits they need to be independent of one another, and from the systems of record

APIs

Are simplest to use, but create a real-time dependency on the underlying system of record

Event streams

Enable microservices to build decoupled views of the data and respond to real time events

http://ibm.biz/eda-resurgence-slides

Comparing messaging and events

Critical data exchange ← → Event driven ← → Event streaming ← →

- Assured Delivery
- Fine grained messaging
- Topics and Subscriptions
- Scalable Subscription
- Stream History

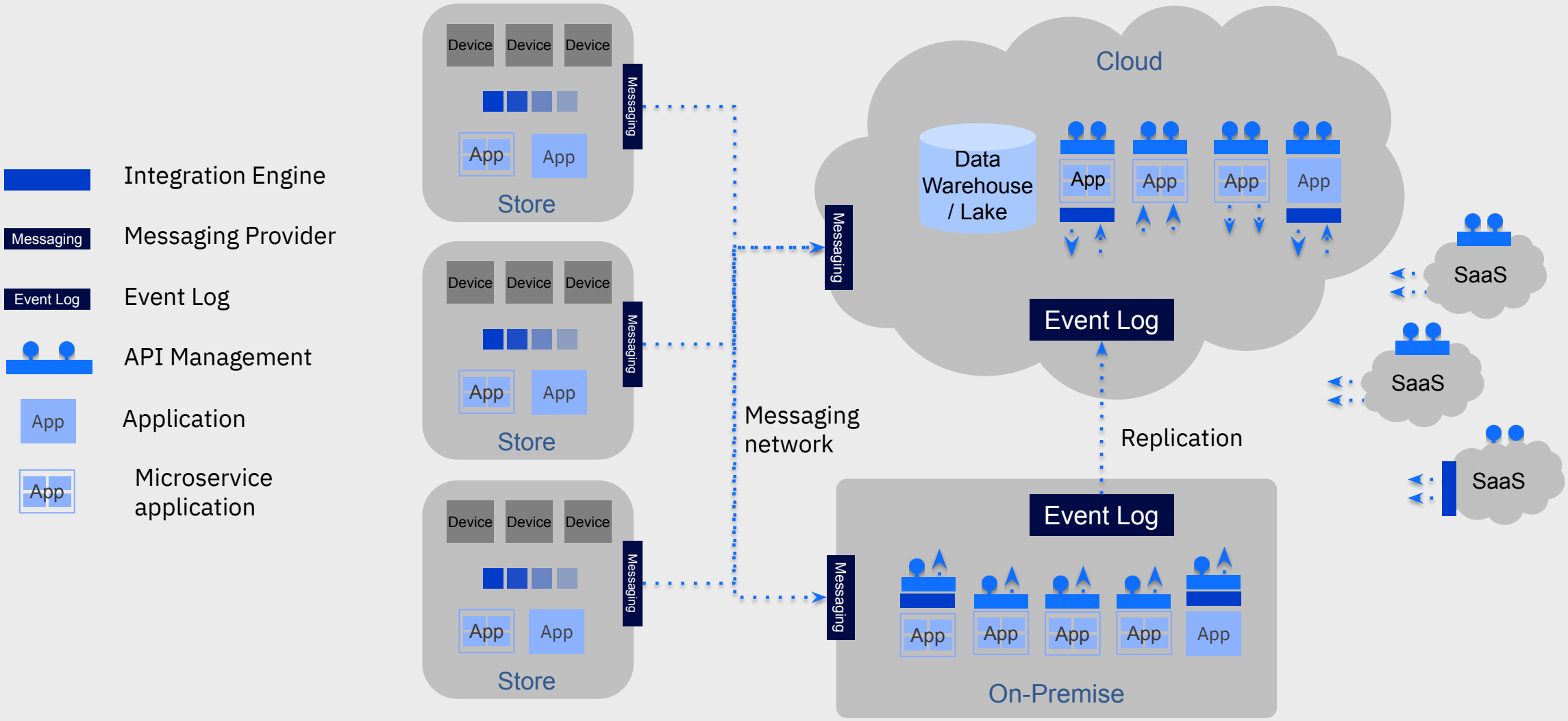
Messaging
Focused on message exchange and transactions

Supported connectivity

Events (e.g. Kafka)
Focused on streaming of events

Strengths ← →

Example of complementary use of messaging and events



Event driven architecture

patterns that build on one another

Event stream data distribution: Event distribution with history

Event stream projections: Consumer specific data views

Event sourcing: Using an event log as a data master

CQRS: Command Query Responsibility Segregation

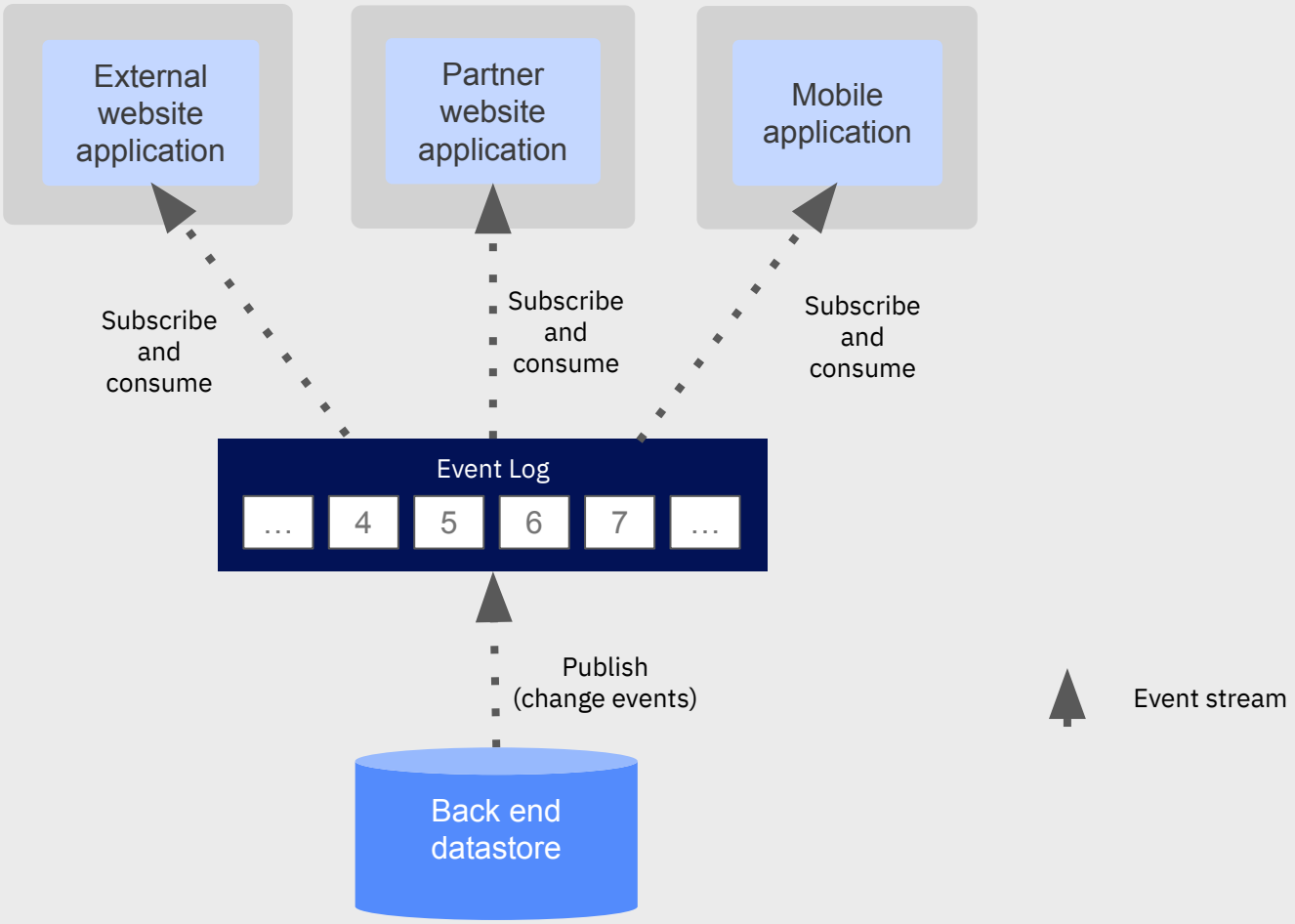
Event processing: Evaluating events over time

Saga: Combining multiple actions together

Supporting/related patterns:

change data capture, event connectors (source/sink), function as a service…

Event streams for data distribution

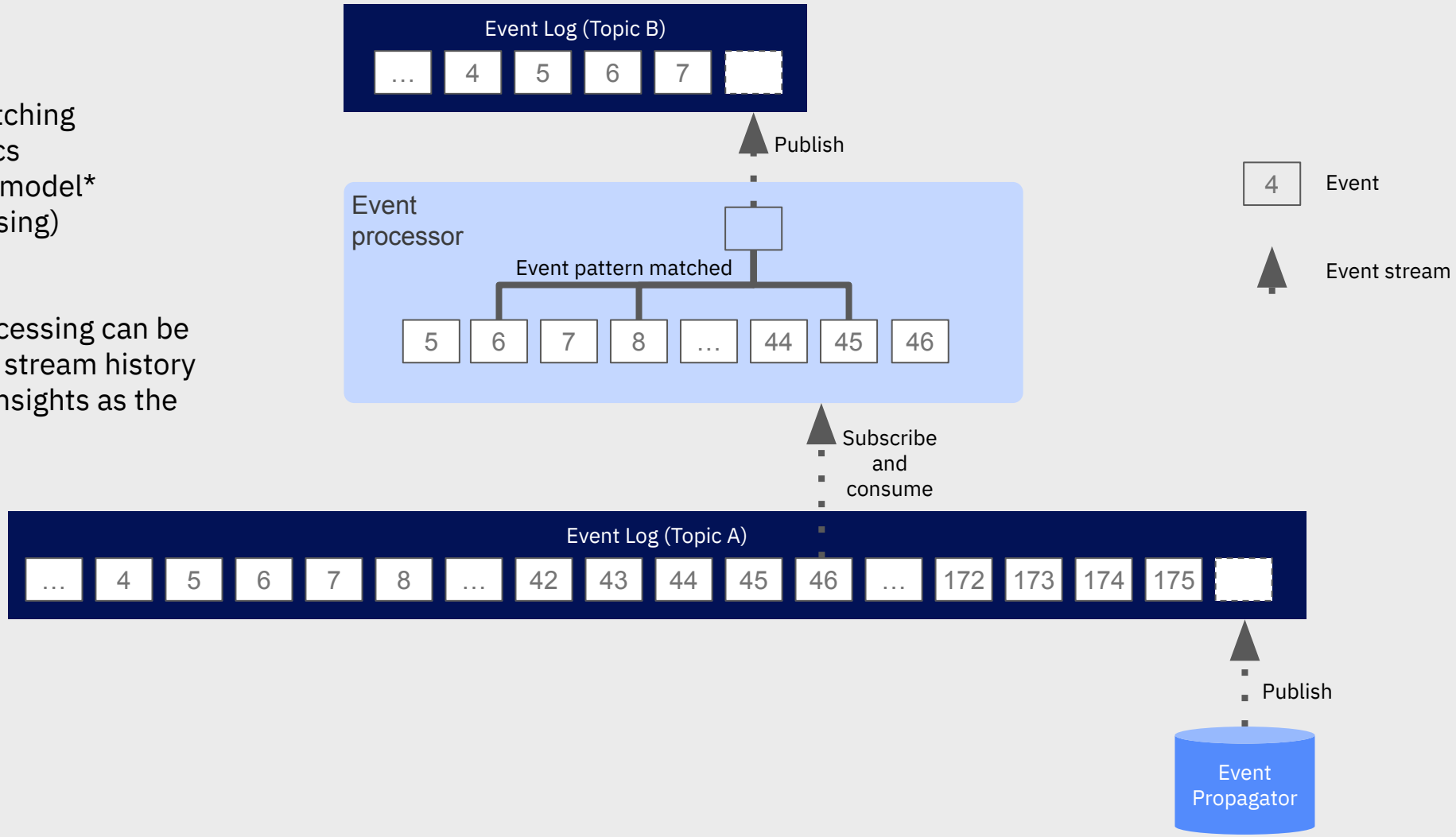


Event processing

Event processing could involve:

- Filtering
- Redaction
- Routing
- Event pattern matching
- Real-time analytics
- Machine learning model* (training and/or using)

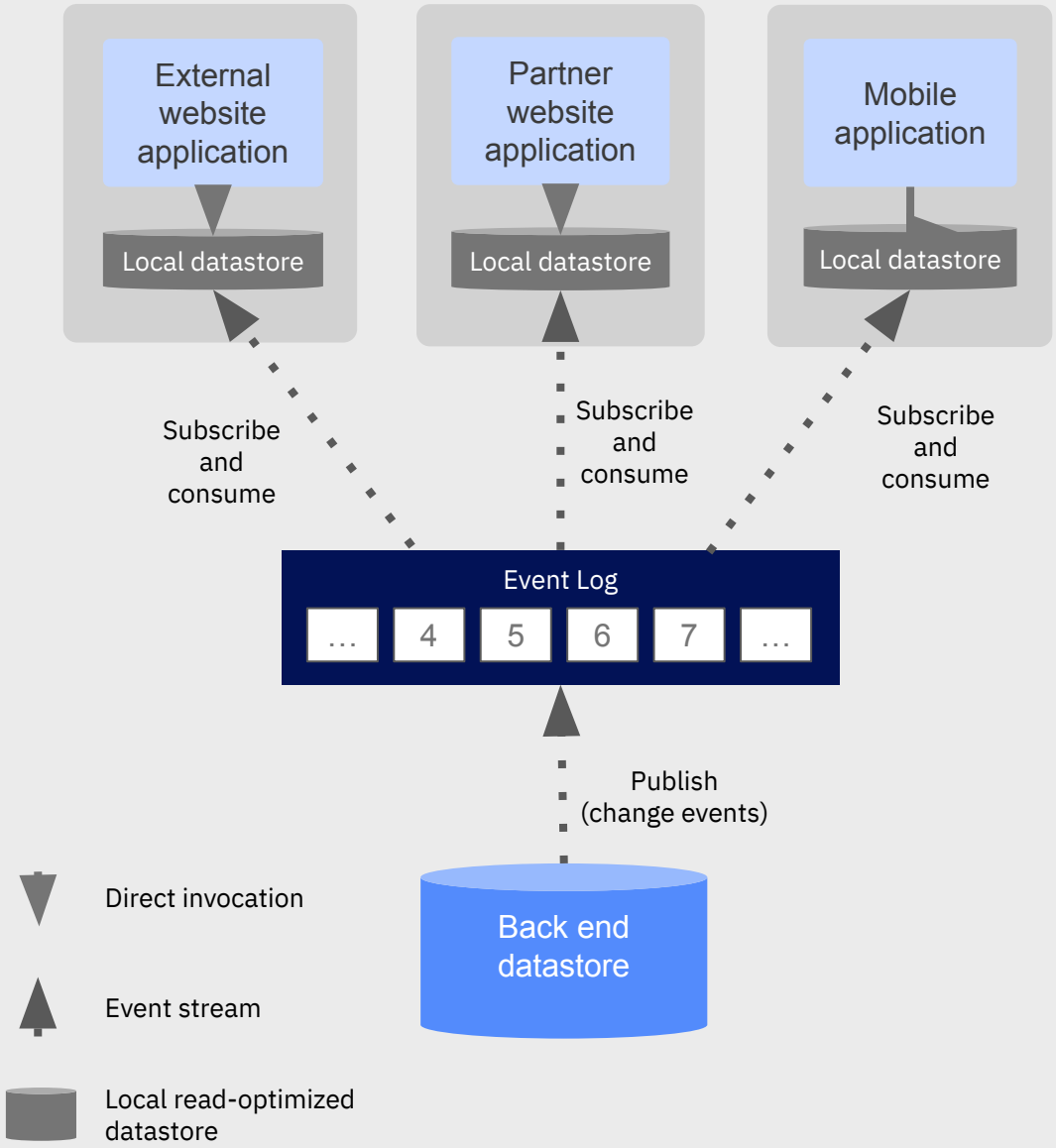
* Note that event processing can be re-run over the event stream history in order to gain new insights as the learning improves



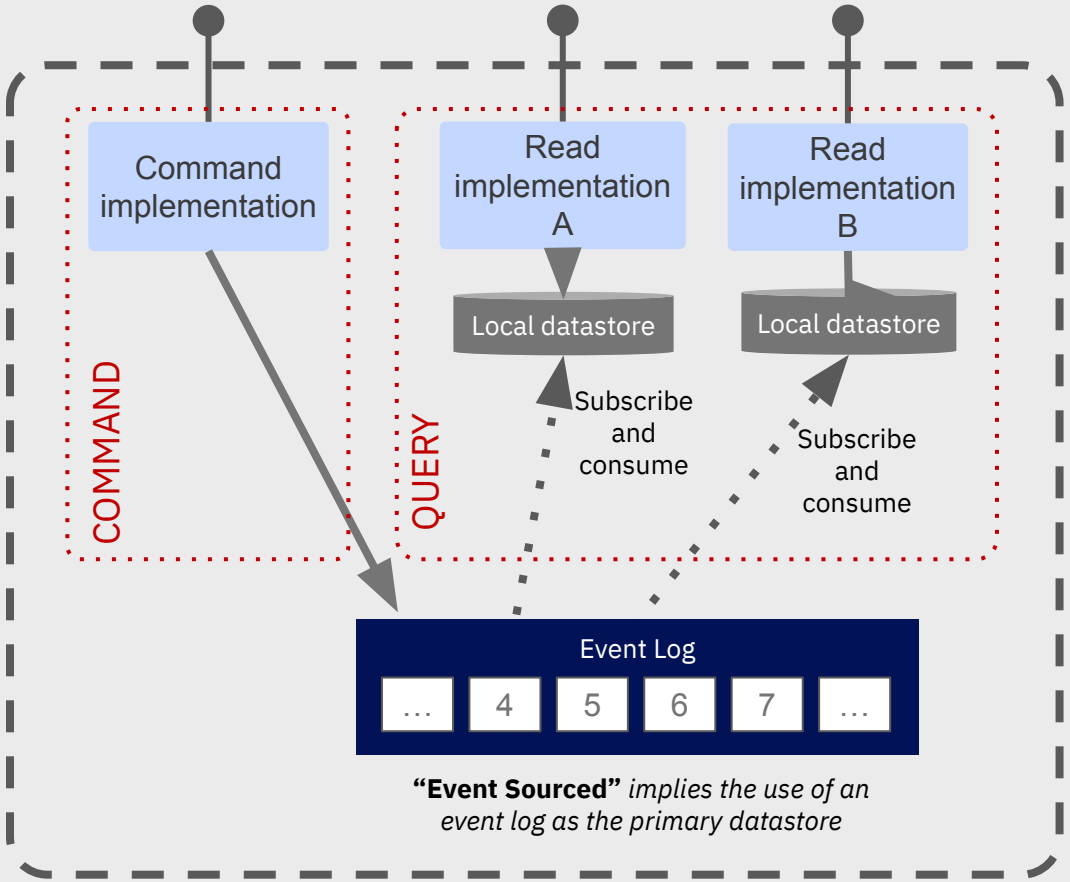
Event stream projections

- Back-end datastores provide a stream of events as changes to their data occur, using *change data capture* or other techniques.
- Those events are published to an event log (such as Kafka) in order to provide a topic-based event stream for applications to listen to.
- These event streams are consumed by applications, to populate their own local data store, with the data optimized into the form they need it.
- These applications' user interfaces (UIs) can query their local data store rather than putting pressure on the back end data store.

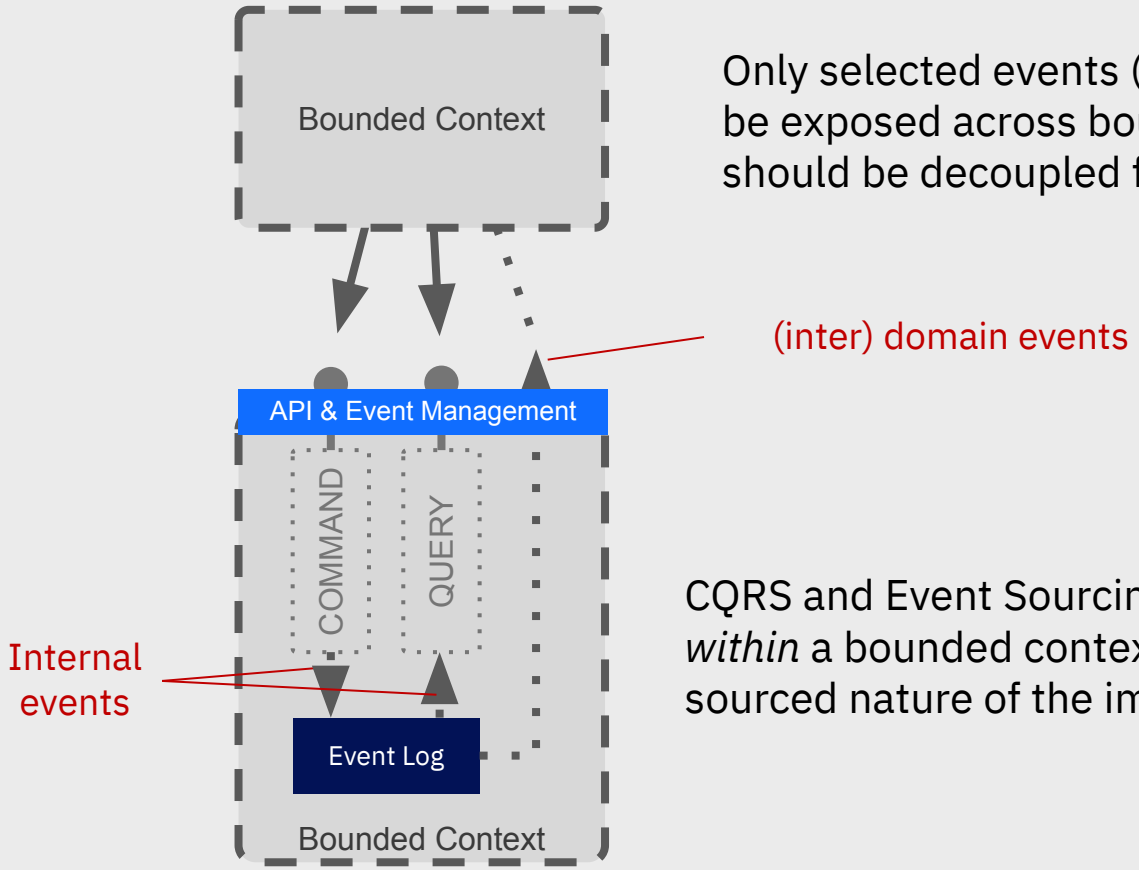
https://developer.ibm.com/articles/event-stream-projections



Event sourcing, and CQRS



Event sourcing/CQRS within a bounded context vs (inter) domain events



Only selected events (and for that matter APIs) should be exposed across bounded contexts. Their data models should be decoupled from the internal model.

CQRS and Event Sourcing patterns are typically scoped to *within* a bounded context. From outside the context, the event sourced nature of the implementation should be hidden.

The maturing space of event endpoint management

Delivered in time to be useful

- Latency measured in seconds not minutes
- Delivered rather than polled

Consumed reliably

- Consumption options suitable for many application types
- Replay history, durable subs, horizontal scale

Easily found, quickly consumed

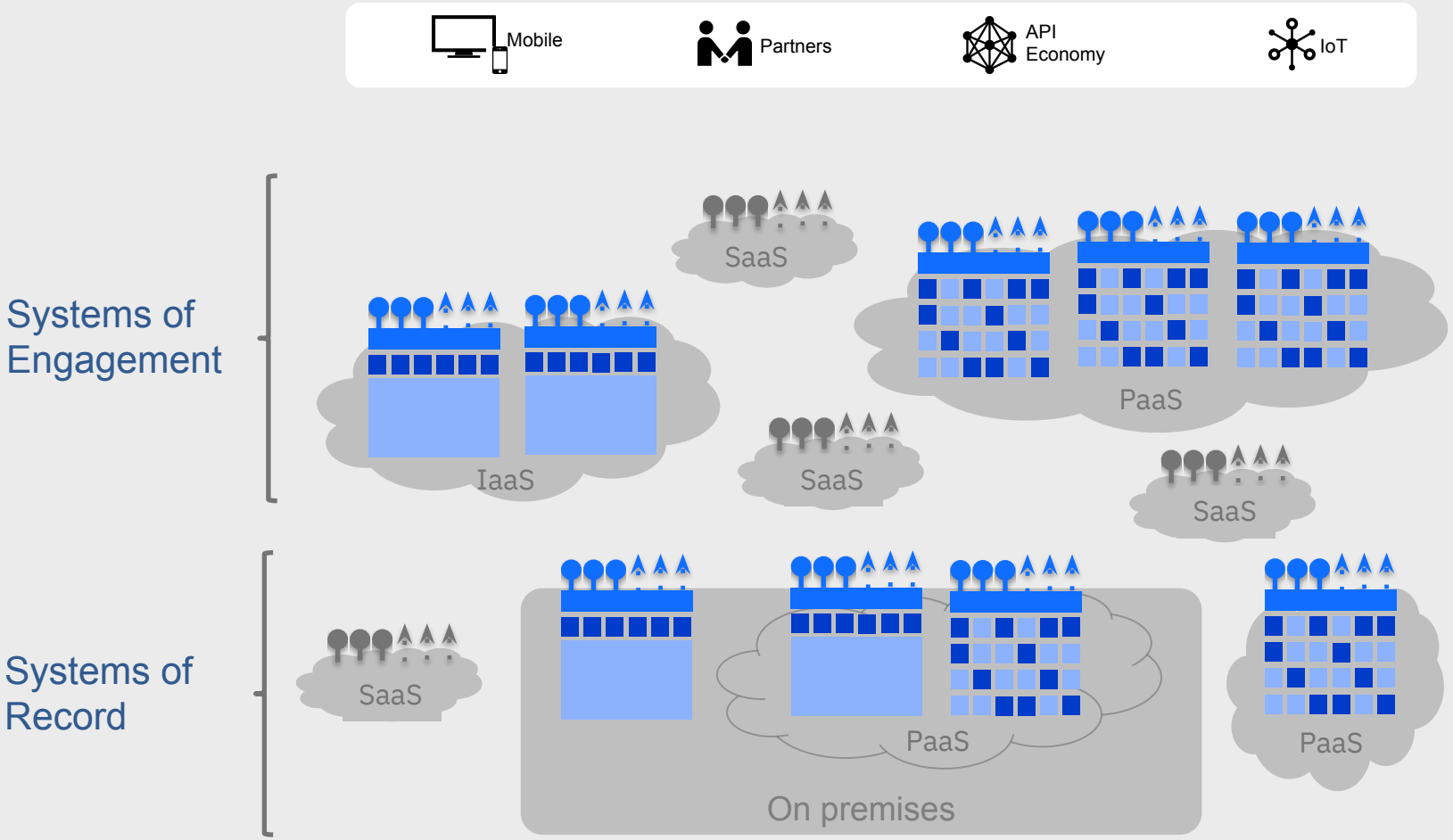
- Described
- Catalogued and discoverable
- Policy controlled access

Completely De-coupled

- Self-service consumption
- No technology ties between provider and consumers
- No operational ties

Event Distribution **Event Endpoint Management**

A **target** high level architecture for enterprise integration



Key features

- Technology layering has been removed.
- Grouping is based on business domains/functions
- Applications are self-contained, including their integration needs.
- Events have become more a first class citizen rather than just a transport
- Well defined application boundaries, breached only by governed APIs and Events
- Cloud platform type (IaaS/PaaS etc.) is hidden when looking from the outside of an application.
- Architecture balances isolation (decoupling), and interaction (defined interfaces).

Caveats

- Represents an idealized *target* integration architecture. Most organizations do not have this level of clarity in the separations of their applications.
- Describes how the things *within* the enterprise communicate with one another. It does not show the enterprise in the context of its broader ecosystem (e.g. with partners).

Exposure points
- APIs
- Events

Polyglot (e.g. microservice application)
- Language runtimes
- Specialist runtimes (e.g. integration)

SaaS

Cloud platform
- SaaS = Software as a service
- PaaS = Platform as a service
- IaaS = Infrastructure as a service

CONFLUENT

Data In Motion

Or how I learned to build a complete streaming app with four simple SQL statements in ksqlDB.


APACHE
kafka®

The Rise of Data in Motion

Data as a continuous stream of events

80%

Fortune 100 Companies Using Apache Kafka

GoPro stripe airbnb

salesforce fitbit box

WIKIPEDIA The Free Encyclopedia Pinterest UBER

Transforming our customers' apps and data architecture

	Without Event Streaming	With Event Streaming
Auto / Transport	Batch-driven scheduling	Real-time ETA
Banking	Nightly credit-card fraud checks	Real-time credit card fraud prevention
Retail	Batch inventory updates	Real-time inventory management
Healthcare	Batch claims processing	Real-time claims processing
Media	Batch data pipelines - production supply chain	Real-time data pipeline
Manufacturing	Scheduled equipment maintenance	Automated, predictive maintenance
Defense	Reactive cyber-security forensics	Automated SIEM and Anomaly Detection

Walmart JPMorgan Chase & Co. DB Morgan Stanley RECURSION pharmaceuticals

Confluent Transforms Data Usage Throughout Enterprises

Retail
Drive consumer analytics & streamline operations

- Inventory Management
- Personalized Promotions
- Product Development & Introduction
- Sentiment Analysis
- Streaming Enterprise Messaging
- Systems of Scale for High Traffic Periods

Healthcare
Provide patients better choices & doctors better insight

- Connected Health Records
- Data Confidentiality & Accessibility
- Dynamic Staff Allocation Optimization
- Integrated Treatment
- Proactive Patient Care
- Real-Time Monitoring

Capital Markets
Combat fraud & remain competitive

- Capital Management
- Early-On Fraud Detection
- Market Risk Recognition & Investigation
- Preventive Regulatory Scanning
- Real-Time What-If Analysis
- Trade Flow Monitoring

Automotive
Amplify vehicle intelligence & safety

- Advanced Navigation
- Environmental Factor Processing
- Fleet Management
- Predictive Maintenance
- Threat Detection & Real-Time Response
- Traffic Distribution Optimization

Common In All Industries
Infrastructure Use Cases

- Data Pipelines
- Messaging
- Microservice/ Event Sourcing
- Stream Processing
- Data Integration
- Streaming ETL

Confluent Customers by Industry

FINANCIAL SERVICES

BANK BRI
Rakuten Card
Bank of America
Credit Suisse
Morgan Stanley
Nordea
EURONEXT
robinhood
Capital One
ING
RBC
JPMorganChase

INSURANCE

CENTENE Corporation
GENERALI
Vitality

TECH

Square
Q2
optimove
DriveCentric

HEALTHCARE

alight
surescripts
RECURSION pharmaceuticals
McKESSON
Humana

COMMUNICATIONS & MEDIA

Telefónica
unity
ticketmaster
sky
Intrado
NETFLIX
Disney+ hotstar
8x8

AUTOMOTIVE/TRANSPORTATION

DB
Lufthansa
BAADER
Advance Auto Parts

CONSUMER/RETAIL

Walmart Save money. Live better.
ao
nuuly
THE HOME DEPOT
Domino's Pizza
Shipt
BOSCH
Boden

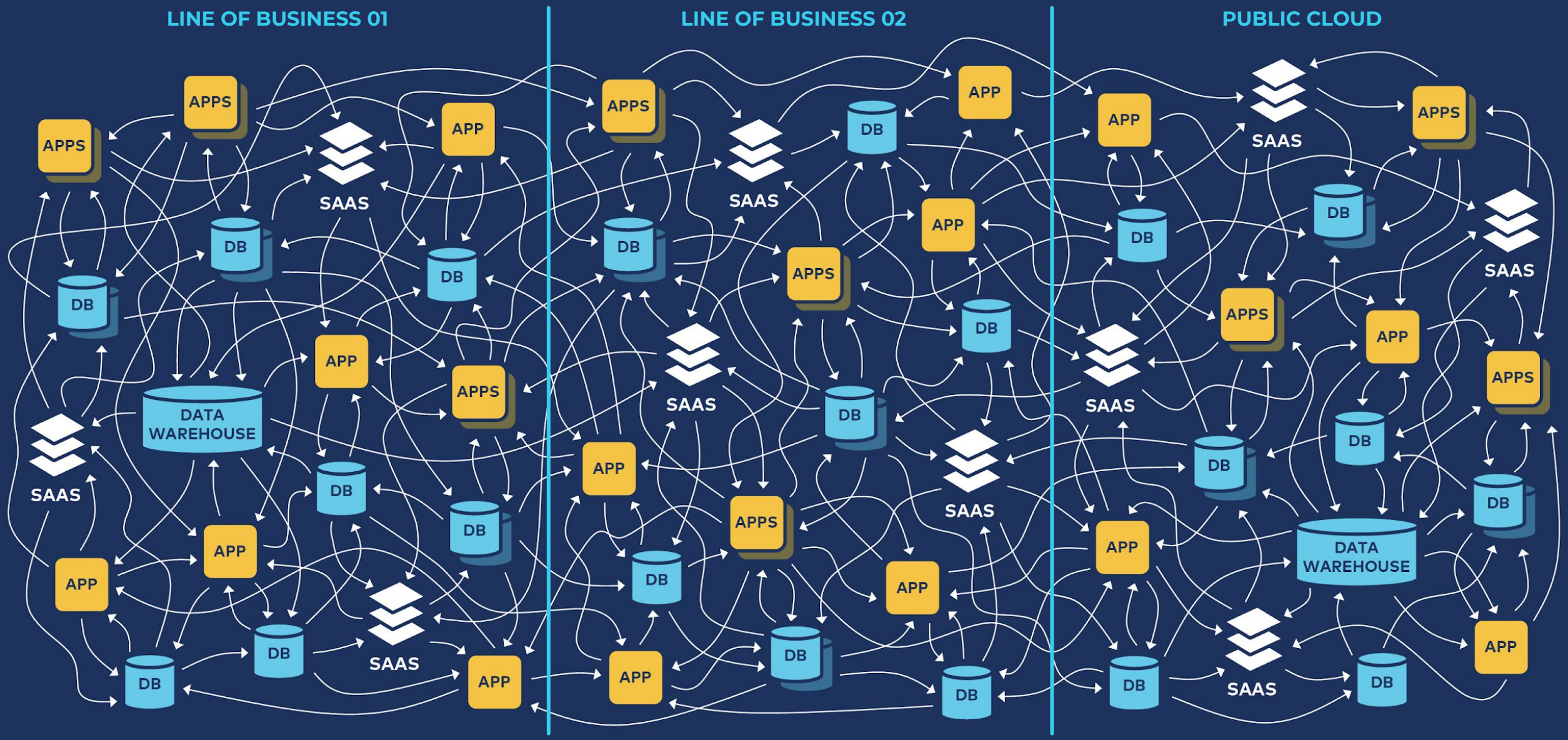
TRAVEL

priceline.com
ACCOR HOTELS
Expedia
HomeAway

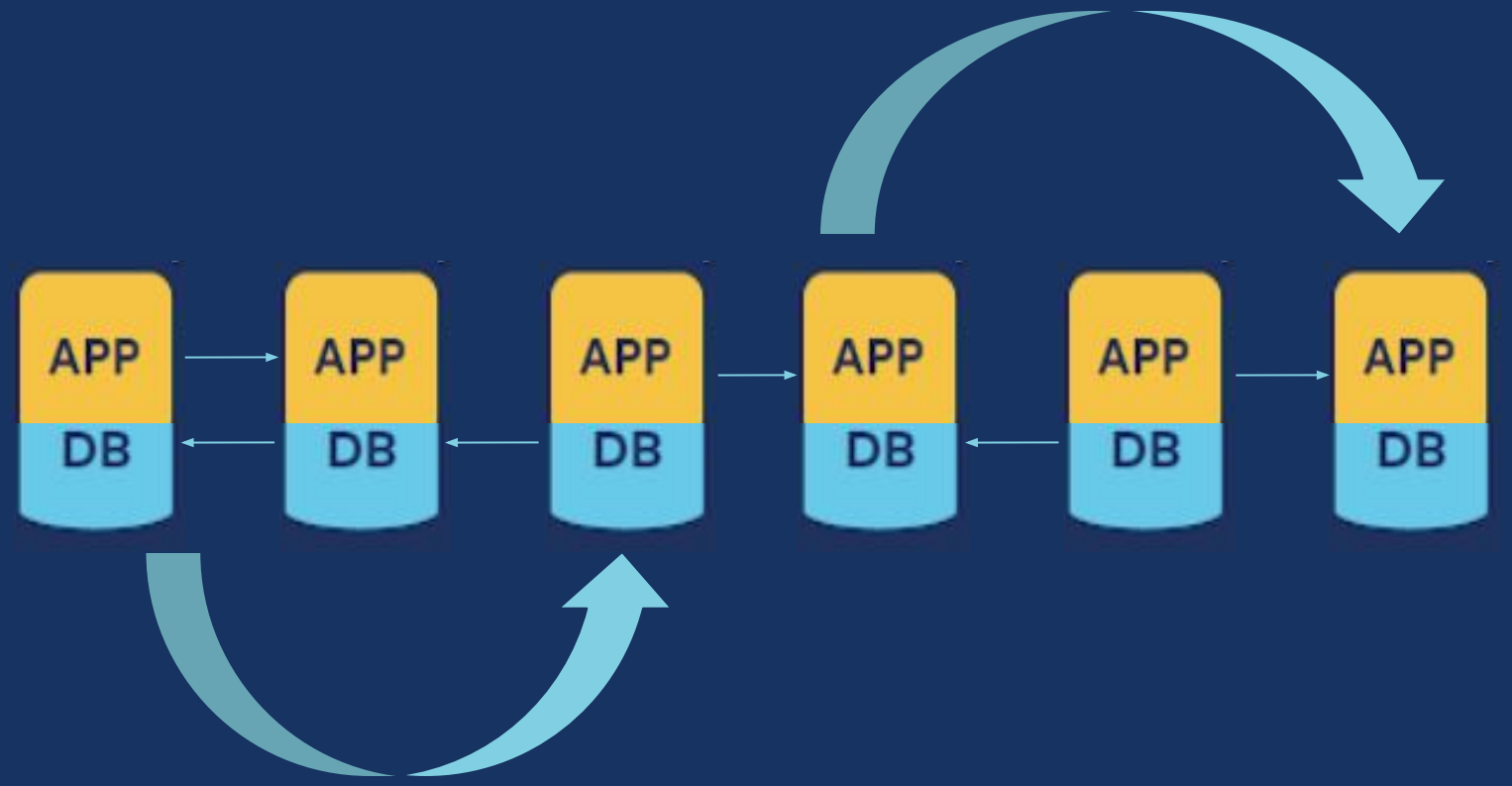
Kafka is powerful … but hard

- Install
- Make secure
- Get data in/out
- Build apps
- Alert errors
- Upgrade
- Configure
- Find data
- Monitor pipelines
- Monitor apps
- Debug
- ?

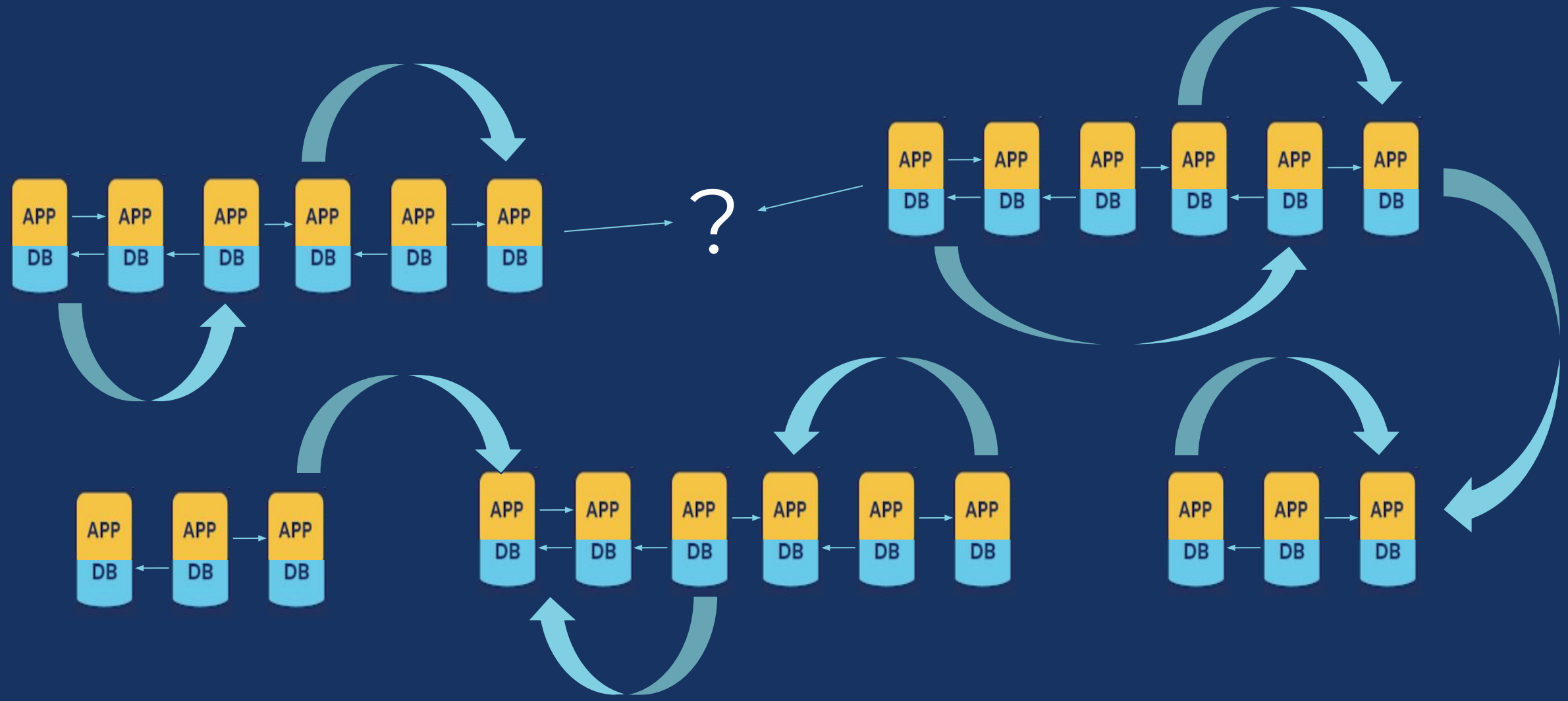
Enterprise Data Architecture is a Giant Mess



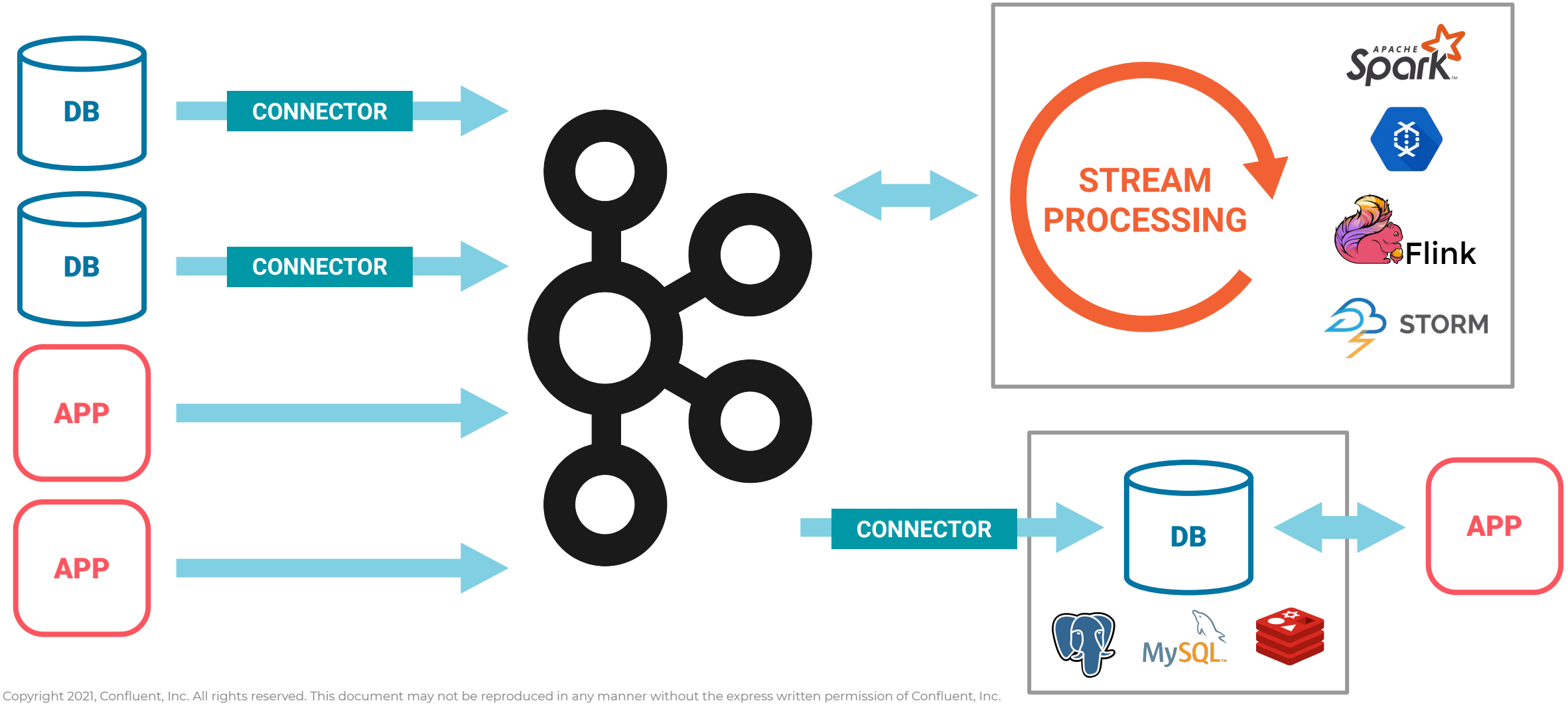
Service Oriented Architecture



Service Oriented Architecture



Most stream processing architectures are complex



Manage

Make changes to Kafka objects and services and see real-time statuses.

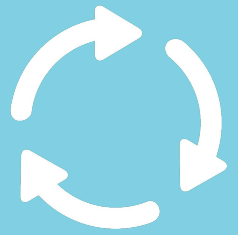
- Create/edit topics
- Change cluster settings
- Manage connectors
- Manage ksqlDB

Monitor

See metrics data for Kafka and connected services over a period of time.

- Broker throughput
- Topic throughput
- Under Replicated Partitions
- Disk usage over

Deploy



Manage Kafka and connected services at scale.

- Upgrade a cluster
- Restart a cluster
- Add a new broker

Confluent Products

DEVELOPER

Unrestricted Developer Productivity

Multi-language Development
Non-Java Clients | REST Proxy

Rich Pre-built Ecosystem
Connectors | Hub | Schema Registry

Event Streaming Database
ksqlDB

OPERATOR

Efficient Operations at Scale

GUI-driven Mgmt & Monitoring
Control Center

Flexible DevOps Automation
Operator | Ansible

Performance & Elasticity
Auto Data Balancer | Tiered Storage

ARCHITECT

Production-stage Prerequisites

Enterprise-grade Security
RBAC | Secrets | Audit Logs

Data Compatibility
Schema Registry | Schema Validation

Global Resilience
Multi-region Clusters | Replicator

Apache Kafka

Open Source | Community licensed

Self-managed Software

Freedom of Choice

Fully Managed Cloud Service

Enterprise Support

Professional Services

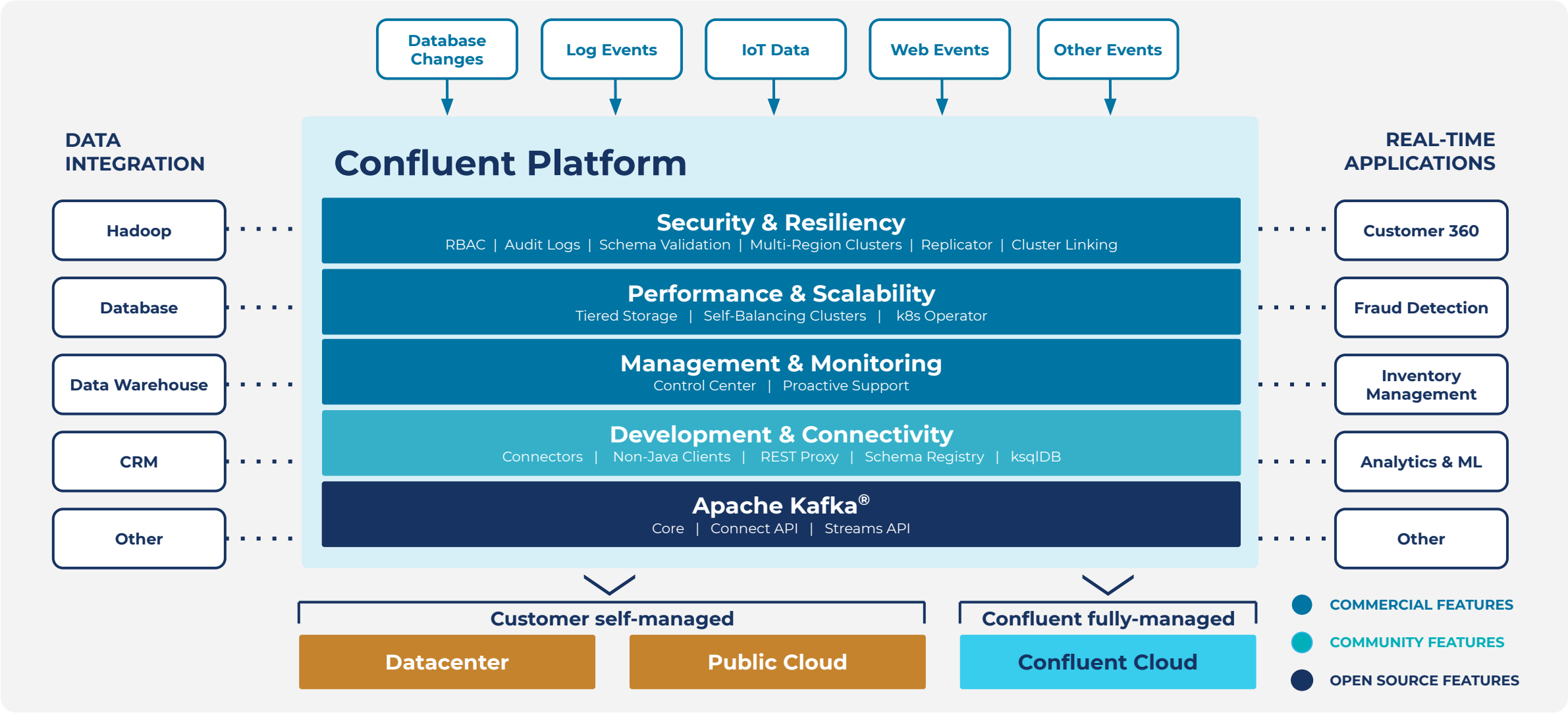
Committer-driven Expertise

Training

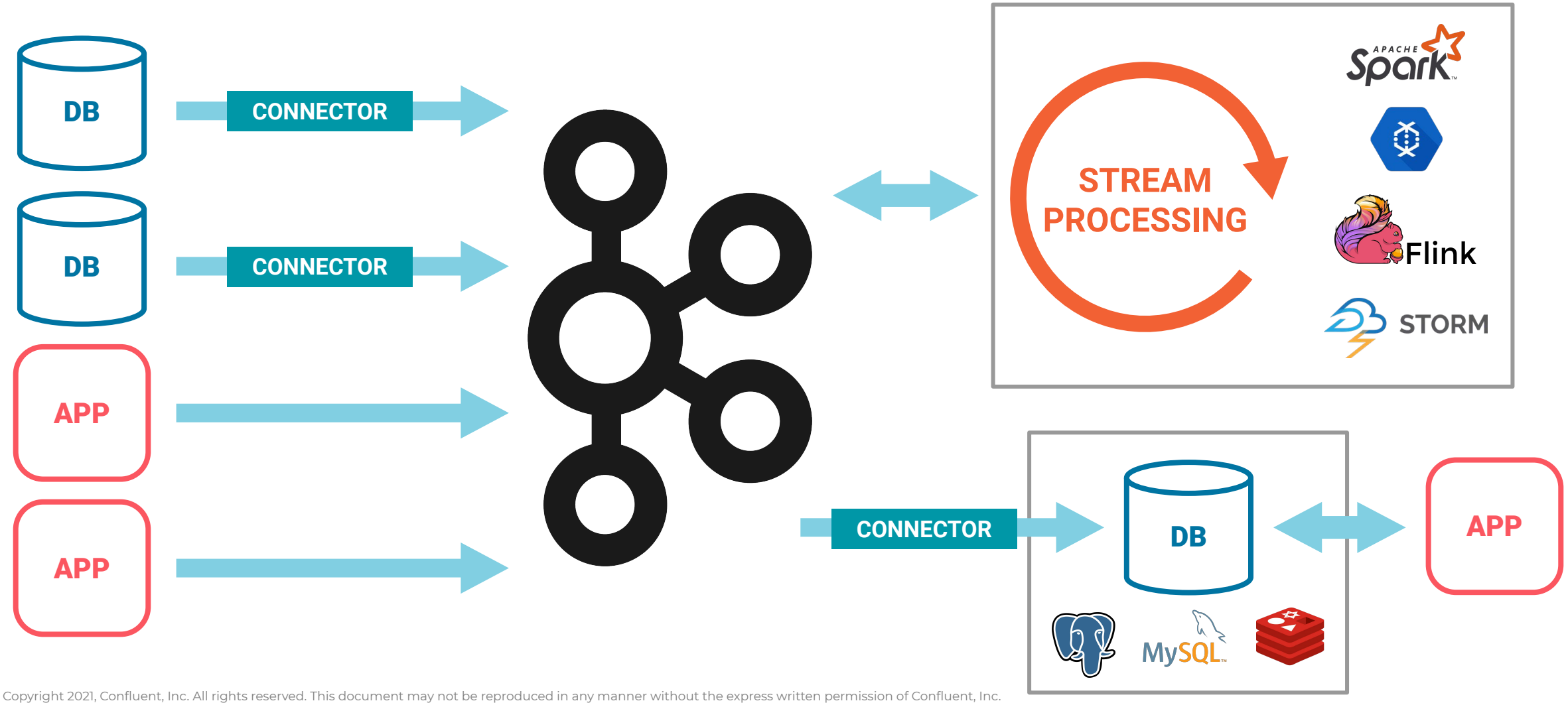
Partners

Complete Technology Ecosystem

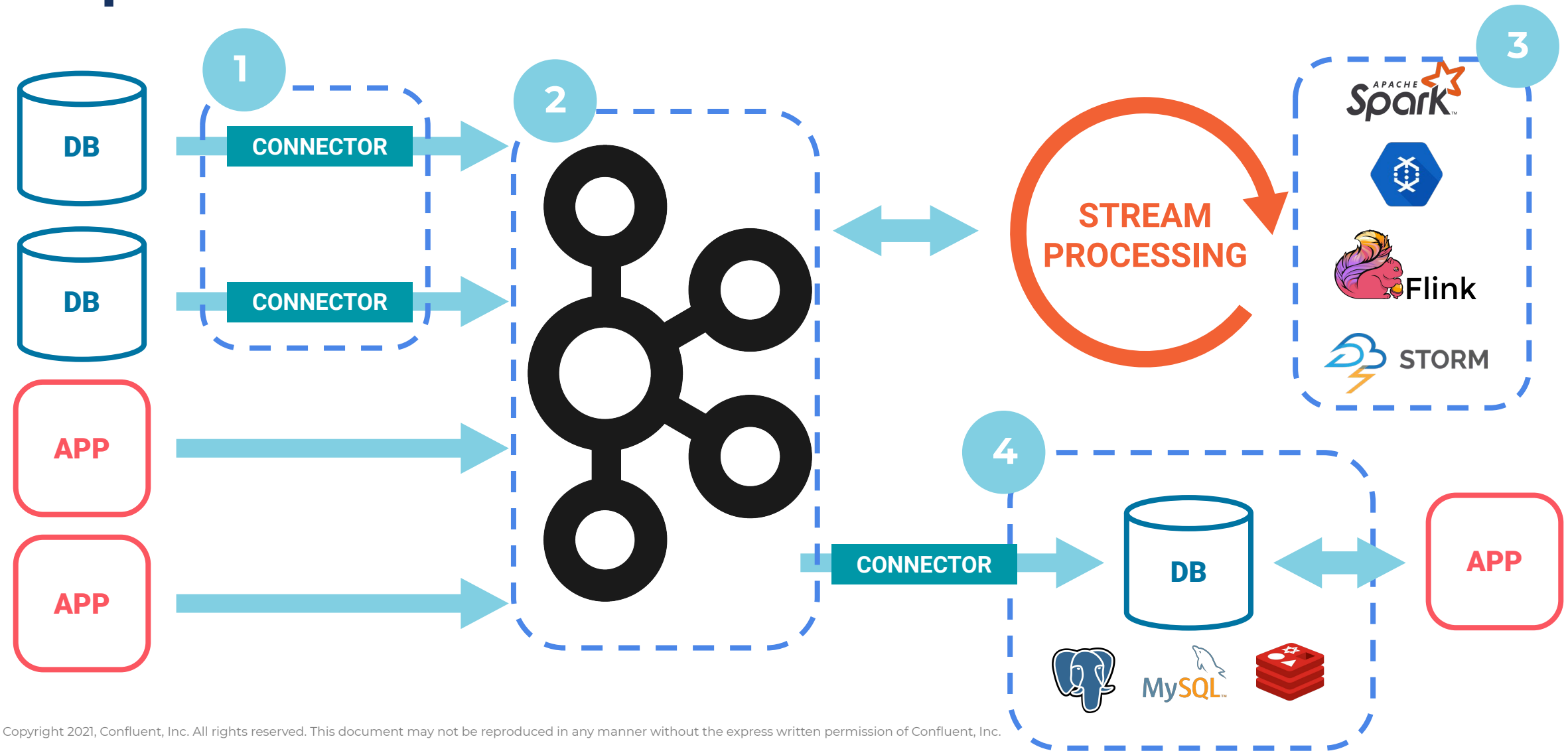
Confluent Delivers A Complete Event Streaming Platform



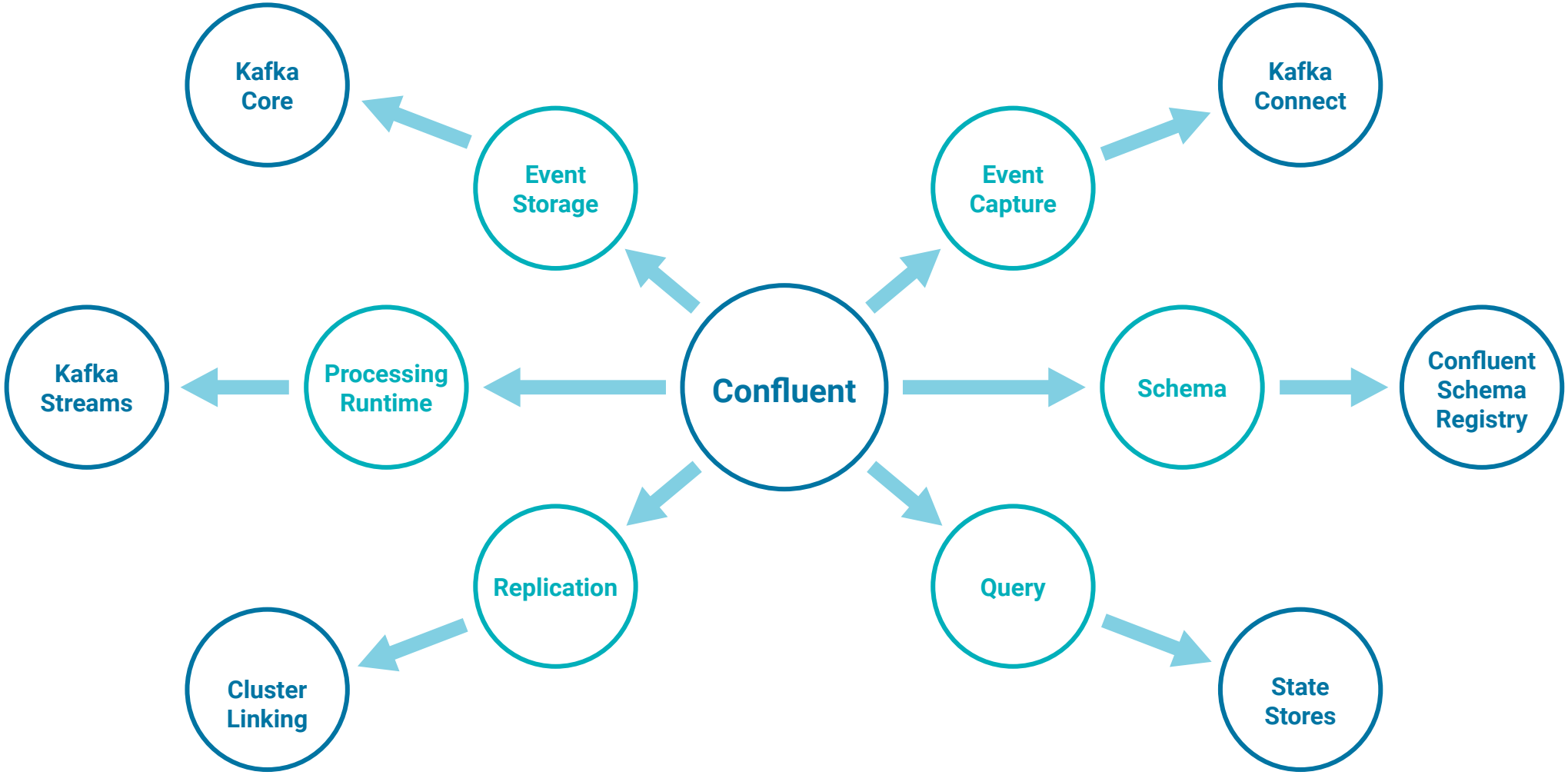
Most stream processing architectures are complex



Most stream processing architectures are complex



Our unfair advantage



Data in motion with Confluent

Kafka producer/ consumer

Kafka Streams

ksqlDB

Stream processing approach comparison

Kafka producer/consumer

```
ConsumerRecords<String, String> records = consumer.poll(100);
Map<String, Integer> counts = new DefaultMap<String,
Integer>();
for (ConsumerRecord<String, Integer> record : records) {
  String key = record.key();
  int c = counts.get(key)
  c += record.value()
  counts.put(key, c)
}
for (Map.Entry<String, Integer> entry : counts.entrySet()) {
  int stateCount;
  int attempts;
  while (attempts++ < MAX_RETRIES) {
   try {
     stateCount = stateStore.getValue(entry.getKey())
     stateStore.setValue(entry.getKey(), entry.getValue() +
stateCount)
     break;
   } catch (StateStoreException e) {
     RetryUtils.backoff(attempts);
   }
 }
}
```

Kafka Streams

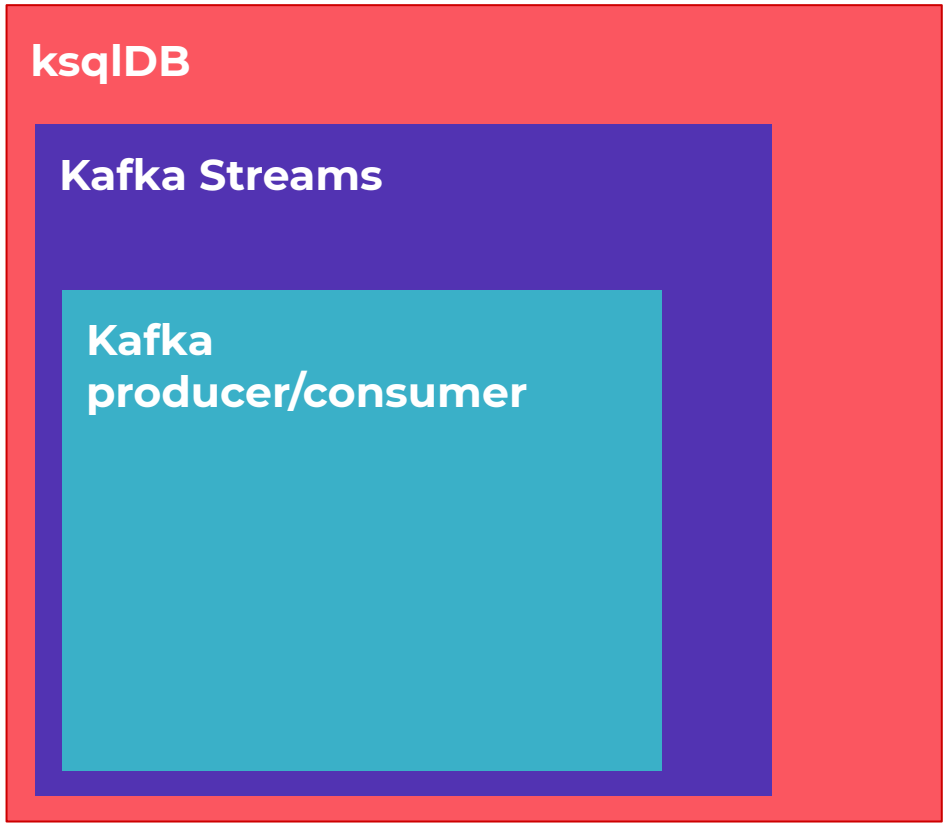
```
builder
    .stream("input-stream",
            Consumed.with(Serdes.String(), Serdes.String()))
    .groupBy((key, value) -> value)
    .count()
    .toStream()
    .to("counts", Produced.with(Serdes.String(), Serdes.Long()));
```

ksqlDB

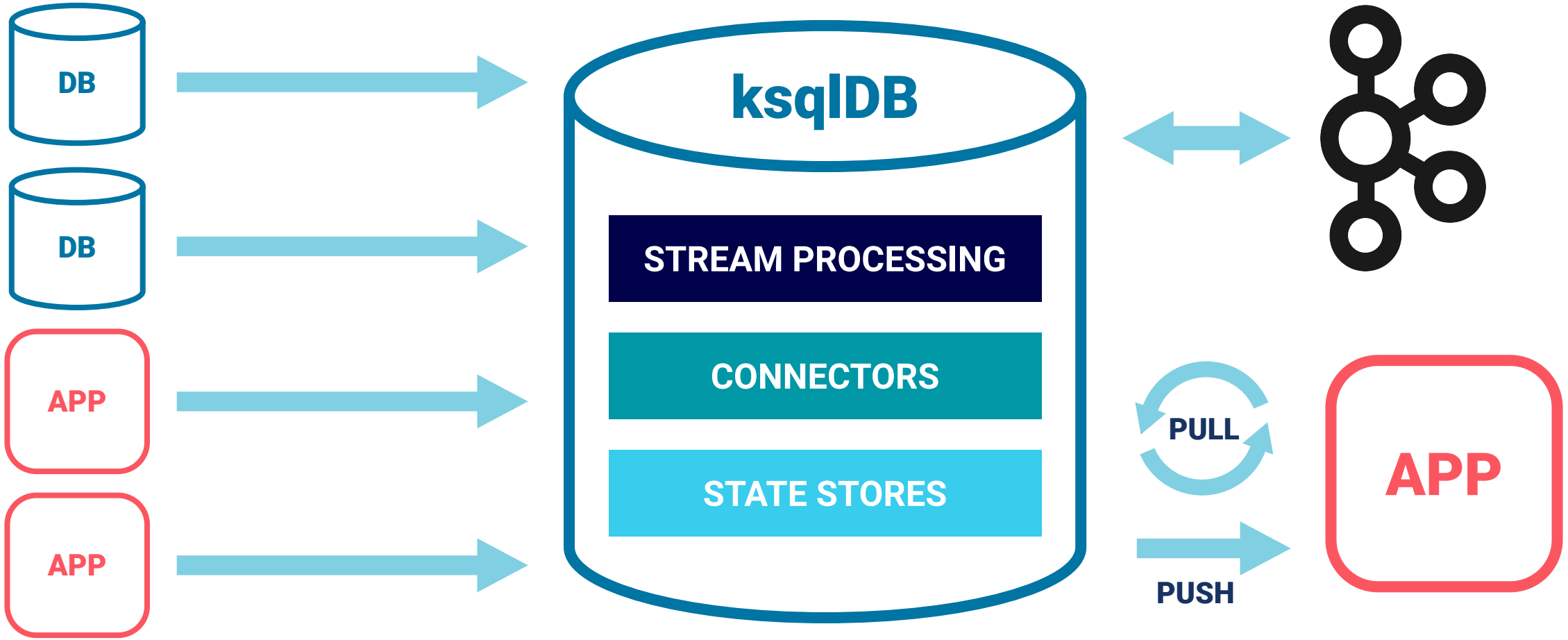
```
SELECT x, count(*) FROM stream GROUP BY x EMIT CHANGES;
```

Stream processing technology organization

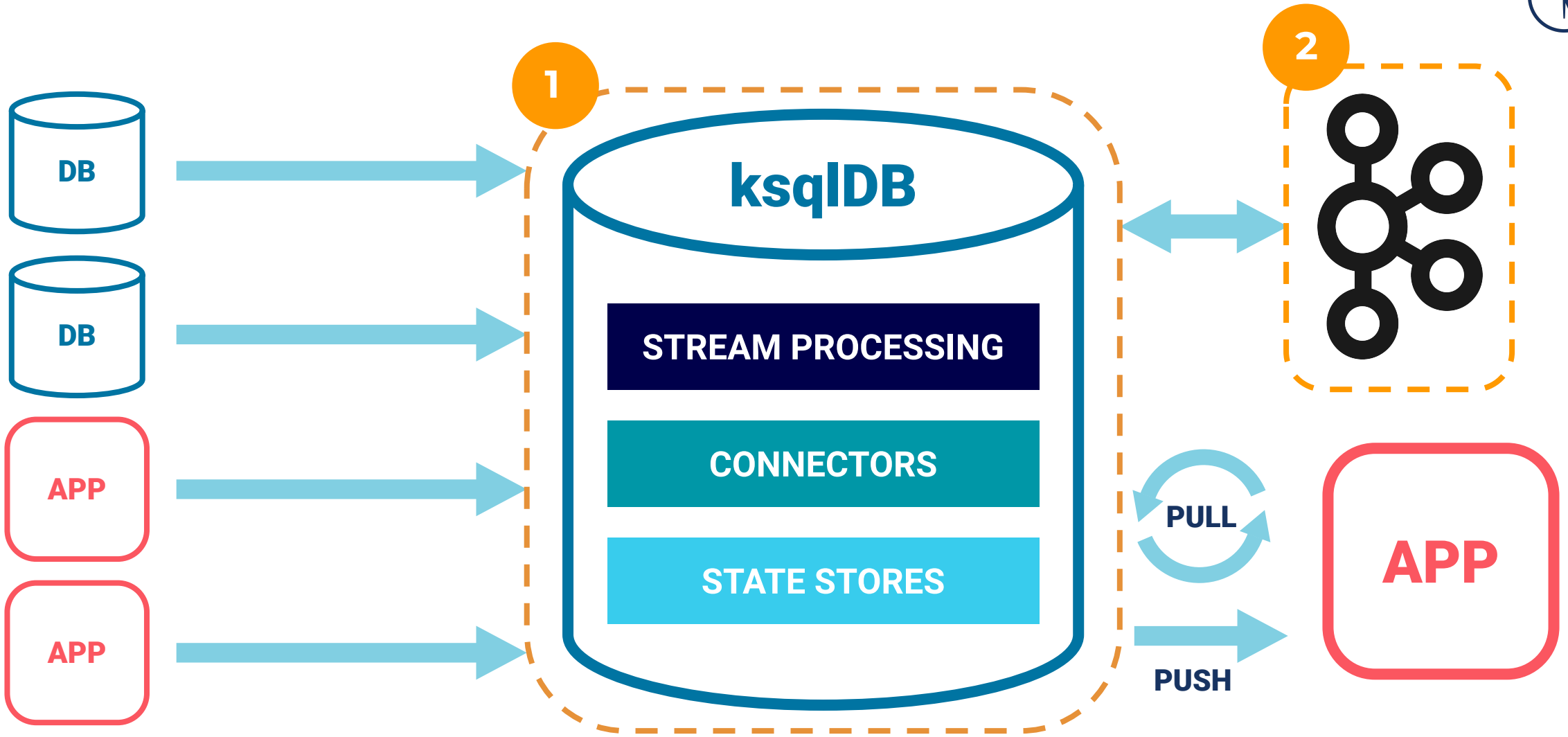
Each layer encapsulates and uses the layer beneath it



An architecture fewer moving parts



An architecture fewer moving parts



Build a complete streaming app with 4 SQL statements

Capture data

```
CREATE SOURCE CONNECTOR jdbcConnector WITH (
  ‘connector.class’ = '...JdbcSourceConnector',
  ‘connection.url’ = '...',
  …);
```

Perform continuous transformations

```
CREATE STREAM purchases AS
  SELECT viewtime, userid,pageid,
         TIMESTAMPTOSTRING(viewtime, 'yyyy-MM-dd HH:mm:ss.SSS')
  FROM pageviews;
```

Create materialized views

```
 CREATE TABLE orders_by_country AS
   SELECT country, COUNT(*) AS order_count, SUM(order_total) AS order_total
   FROM purchases
   WINDOW TUMBLING (SIZE 5 MINUTES)
   LEFT JOIN purchases ON purchases.customer_id = user_profiles.customer_id
   GROUP BY country
   EMIT CHANGES;
```

Serve lookups against materialized views

```
SELECT * FROM orders_by_country WHERE country='usa';
```

CONFLUENT

Learn more

IBM Cloud Pak for Integration

ibm.com/cloud/cloud-pak-for-integration

Confluent

confluent.io

Agile integration

ibm.biz/agile-integration

ibm.biz/agile-integration-webinar

ibm.biz/eda-resurgence

Thank you!

Kim Clark
kim.clark@uk.ibm.com