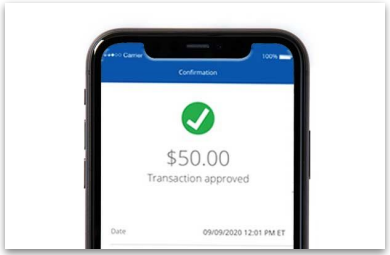
Evolving from Messaging to Data in Motion

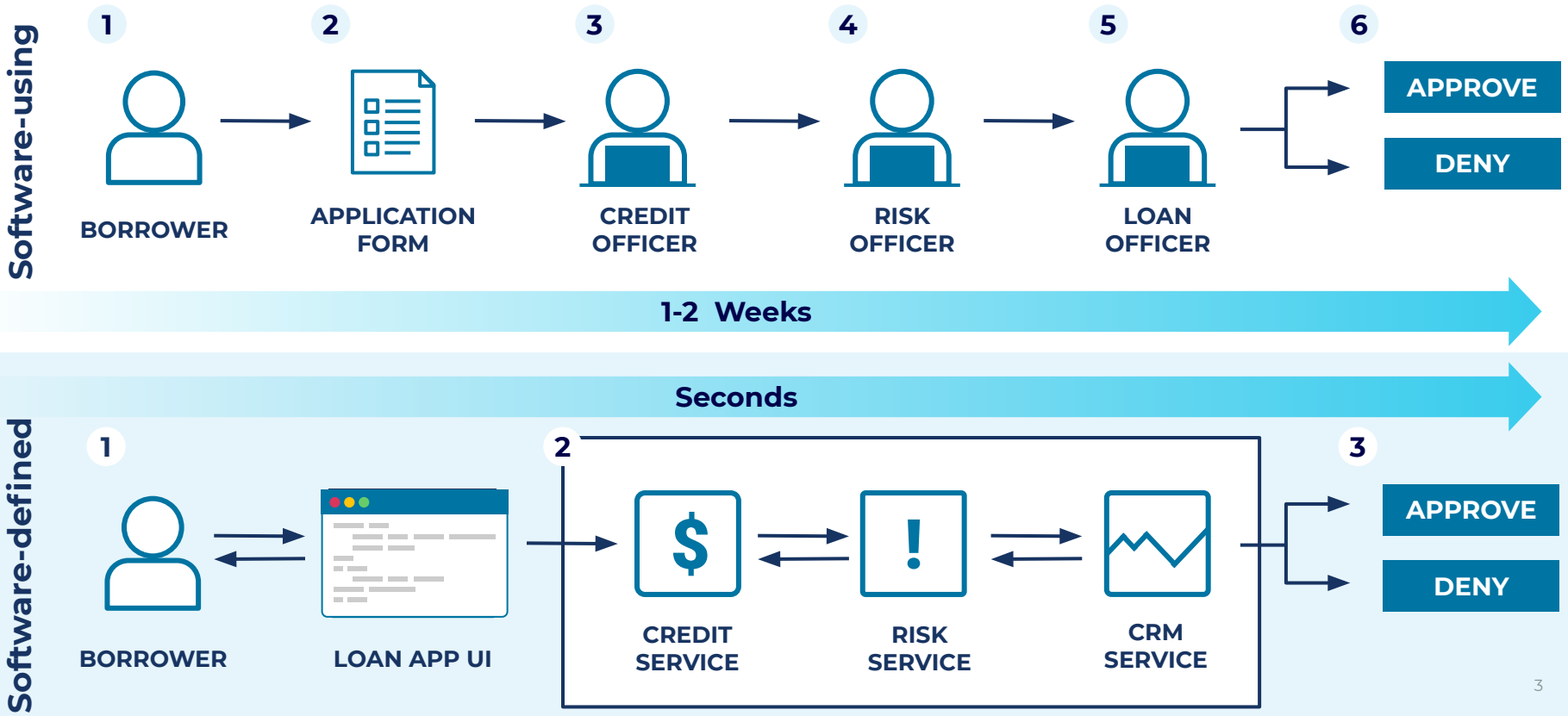
Confluent for Messaging Modernization

Perry Krol, Manager Solutions Engineering CEMEA

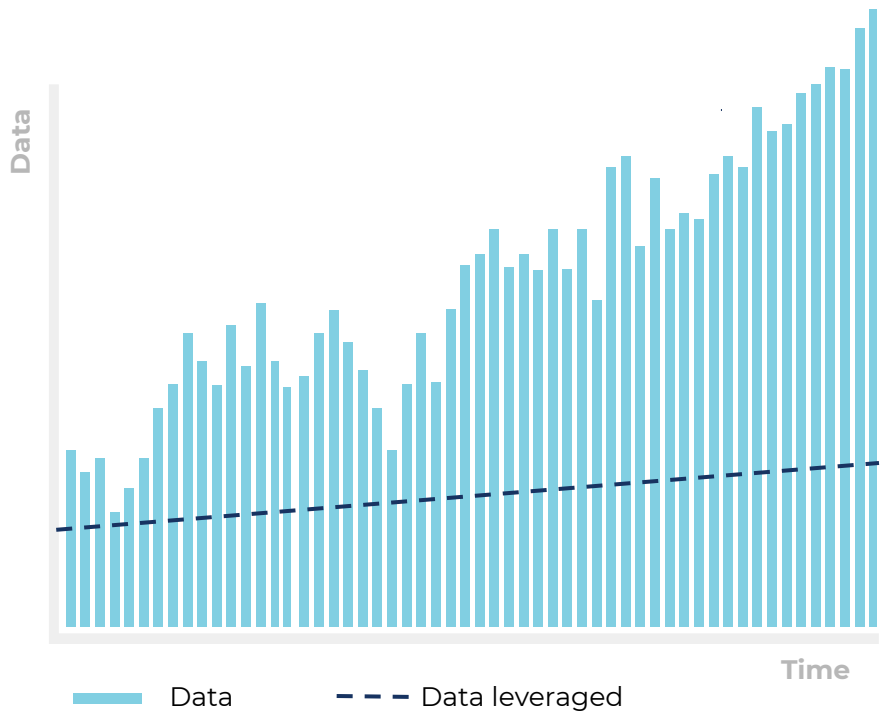
Every Business is Becoming Software

	Banking	Mortgage	Taxi	Grocery
Then				
Now				

Example: Loan Application Process



Unlocking Value from Data, in Real-time, is Imperative to Transformation



- Purchases
- Workflows
- App data
- Security events
- Interactions
- Search data
- IoT and so much more

Employees

Customers

Enterprises are Undertaking Multiple Initiatives to Enable Transformation

Microservices

Cloud

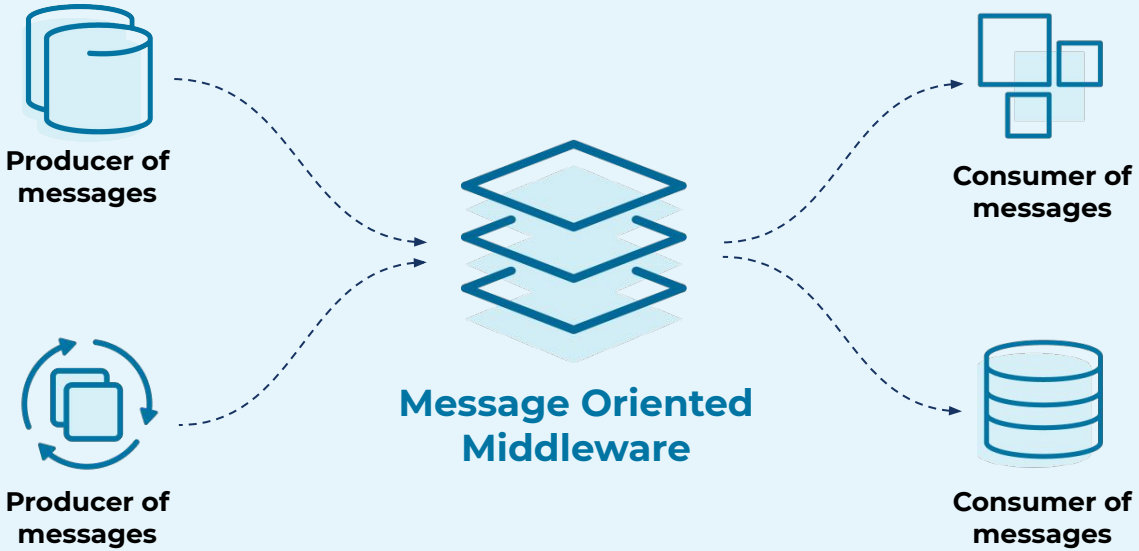
Automation

Machine Learning

The Problem: your use of data has changed but the supporting infrastructure has not!

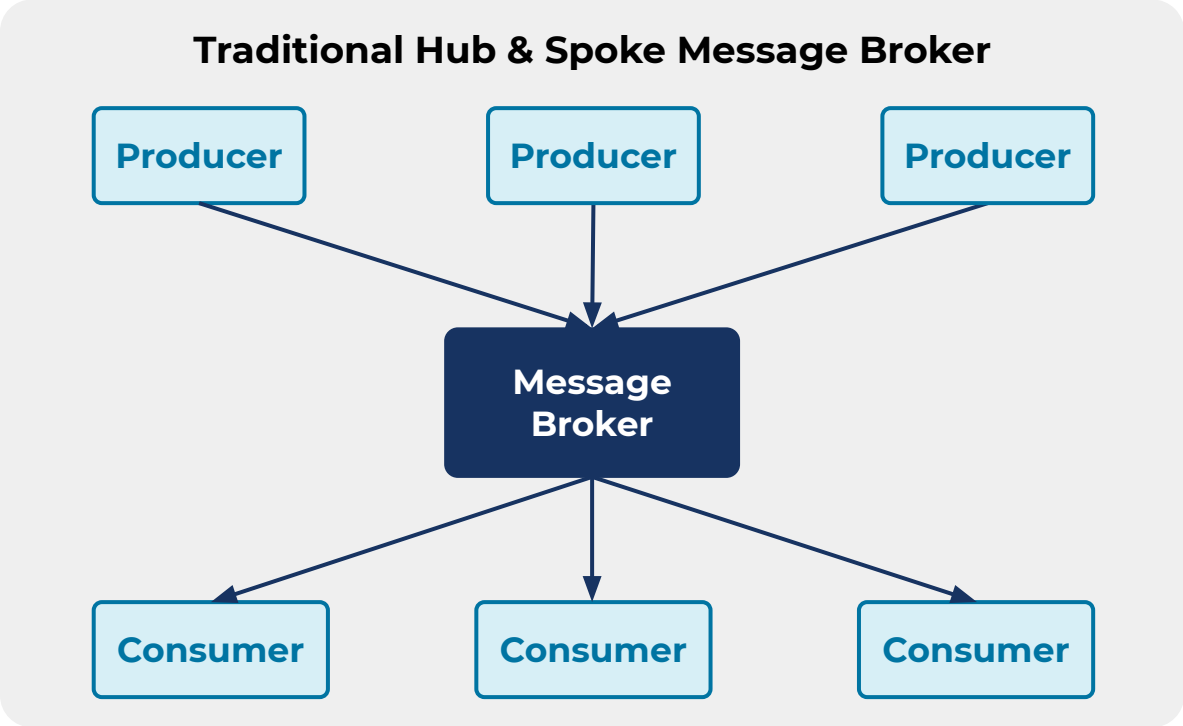
Paradigm for Data-at-Rest: Databases

Paradigm for Data Movement: Messaging Middleware



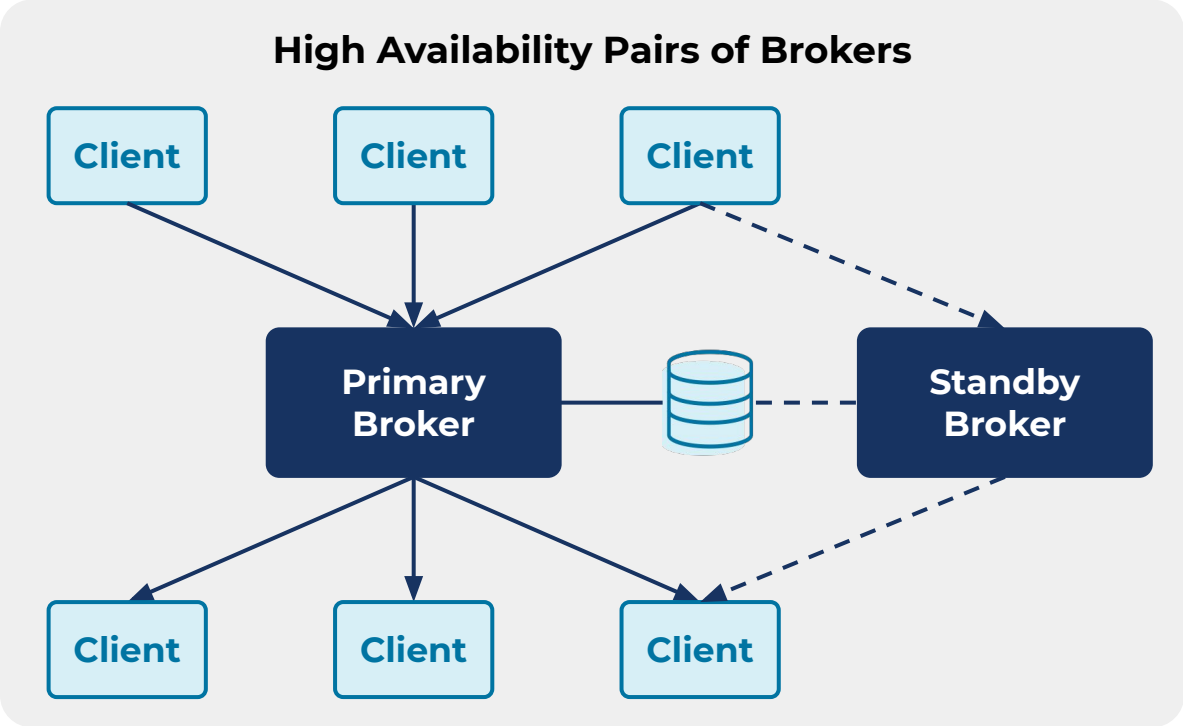
“Message Exchange”

Message brokers were originally architected as centralised systems



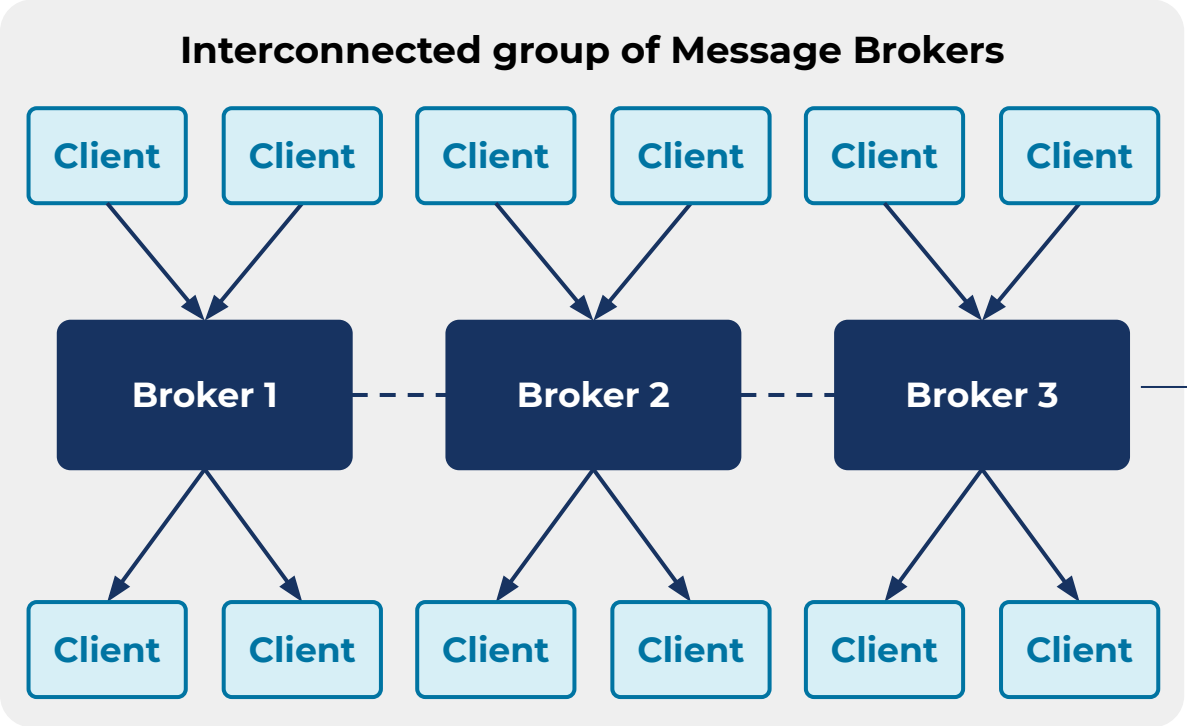
Single point of failure, resulting in **low fault-tolerance and high downtime**

For improved fault-tolerance, brokers are often deployed in high availability pairs



Clients still **only connect to the active brokers** though, so this solution **lacks scalability...**

Over time, some brokers evolved to multi-node networks, but clients still connect to one broker



Because clients still **only connect to one broker**, this solution **still does not provide horizontal scalability**

Managed independently

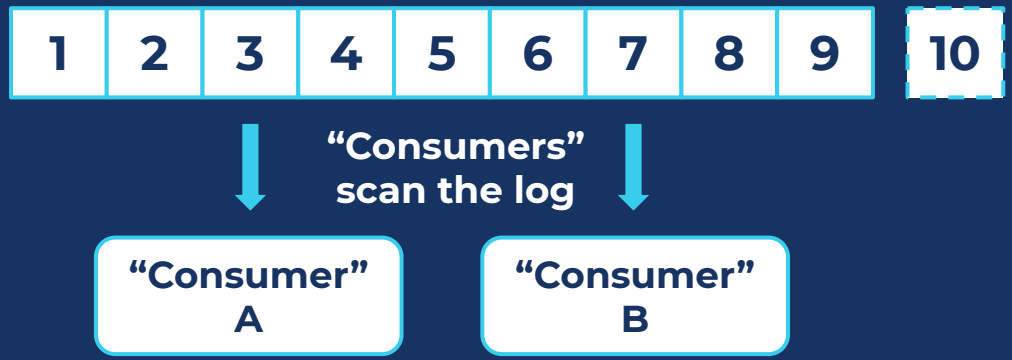
The world is not Ephemeral!

Let’s use an immutable log to share data!

Kafka producers write to an append-only, immutable, ordered sequence of messages, which is always ordered by time

- Sequential writes only
- No random disk access
- All operations are O(1)
- Highly efficient

A log is like a queue, but re-readable :-D

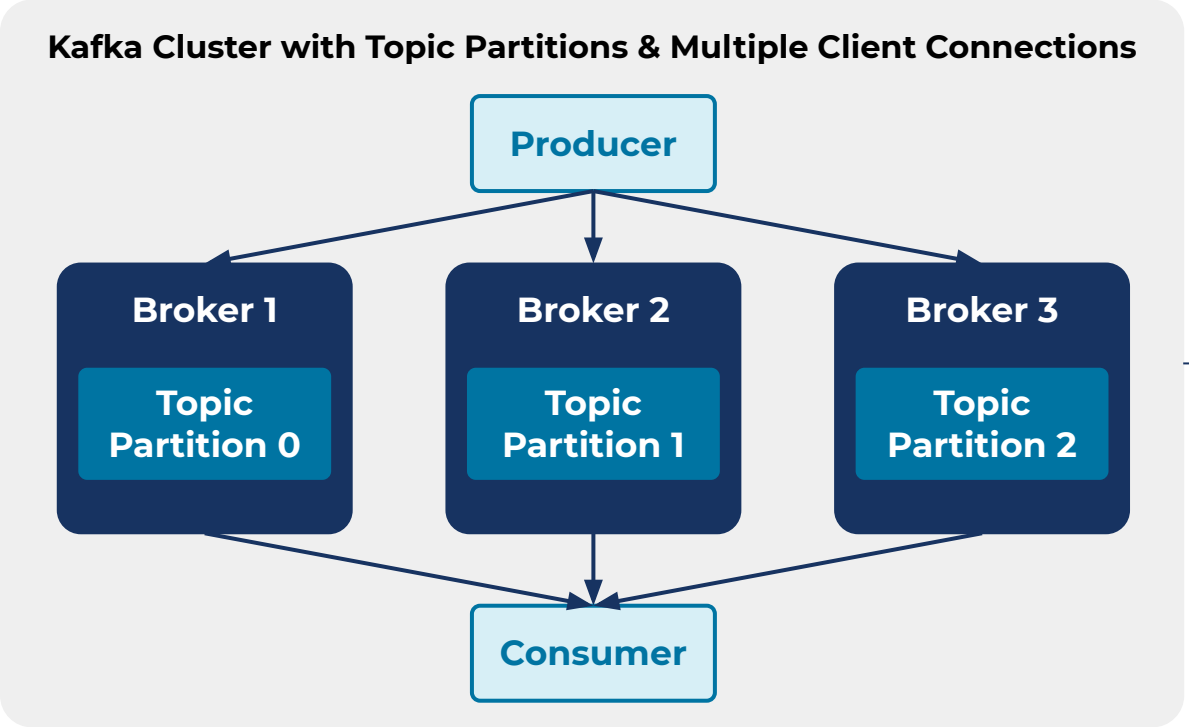


"**Better than a queue**"-like behavior as Kafka consumer groups allows for parallel in-order consumption of data, which is something that shared queues in traditional message brokers do not support.

- Sequential reads only
- Start at any offset
- All operations are O(1)
- Highly efficient

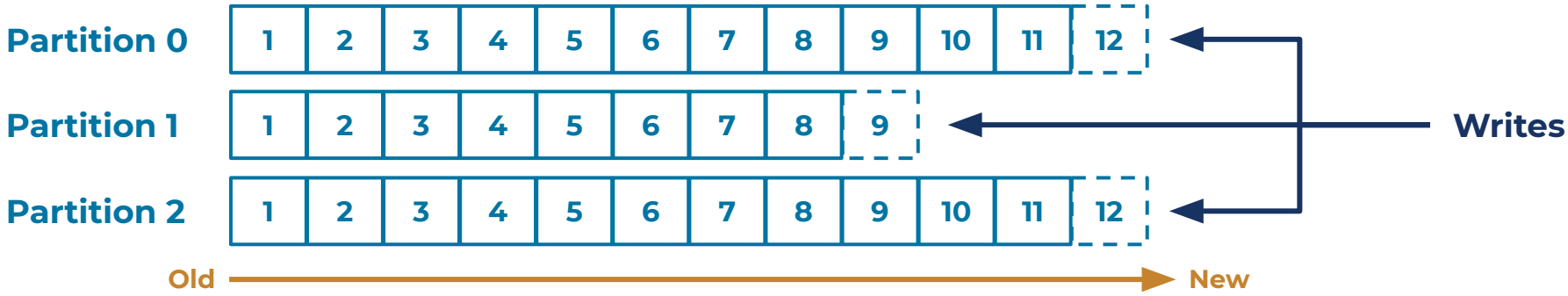
Slow consumers don't back up the broker: **THE STREAM GOES ON.**

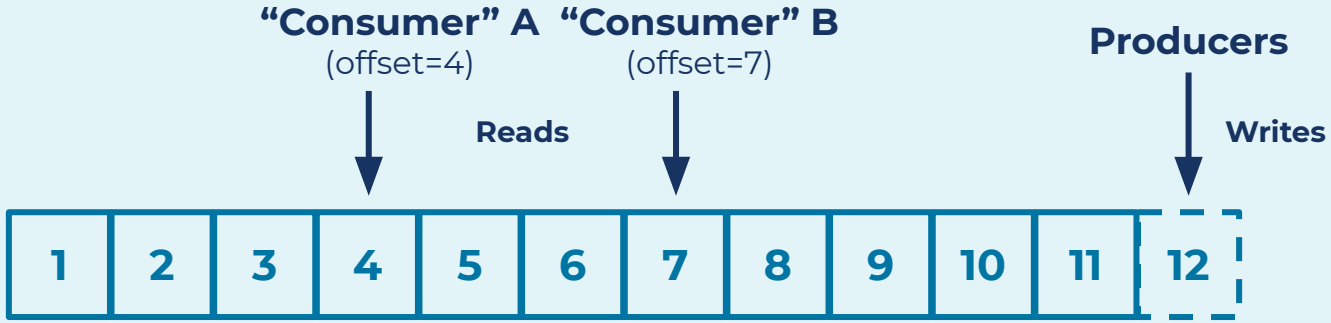
Kafka takes a different approach by partitioning topics across the brokers in a cluster



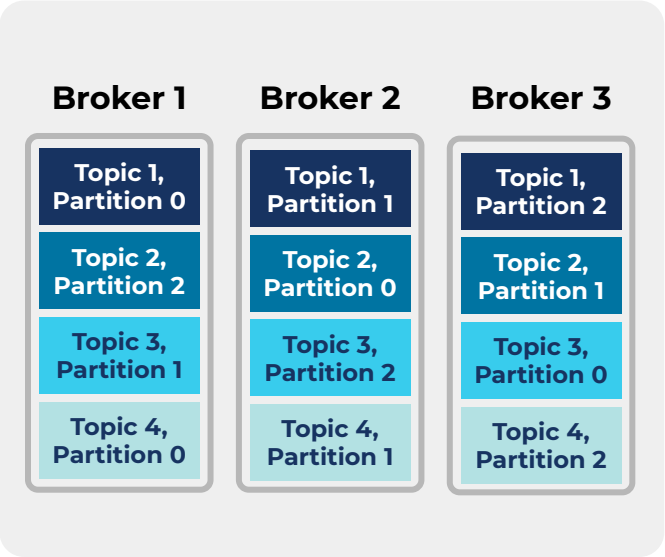
Clients connect to multiple brokers for both reads and writes to and from a topic

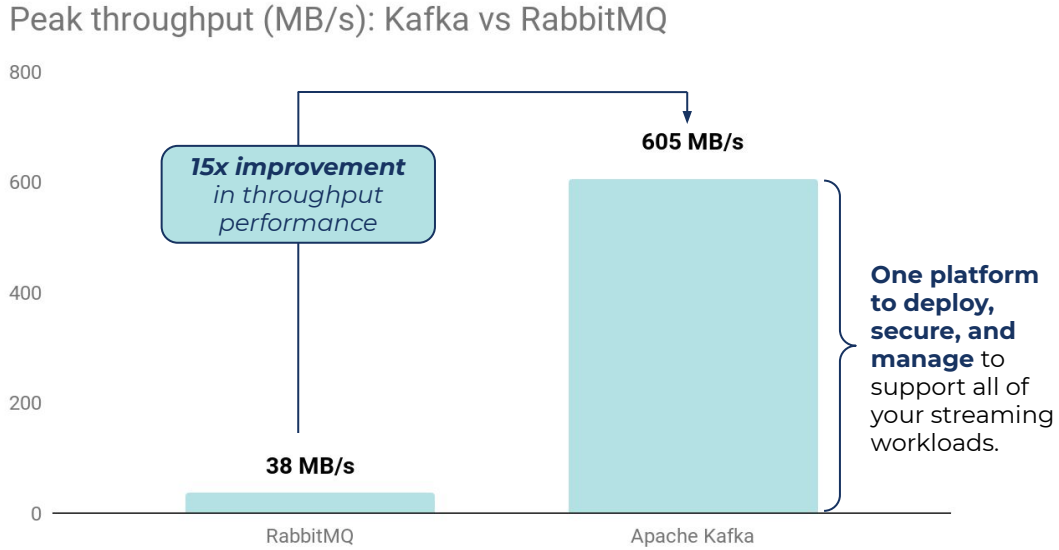
Kafka topics are designed as a commit log that captures events in a durable, scalable way





Partitioning topics enables greater horizontal scalability and enterprise-scale throughput





How else is Kafka different from traditional messaging queues?

Topic partitions are replicated to maximize fault-tolerance

In addition to partitioning topics, each partition can be replicated across multiple brokers to **ensure high uptime even if a broker is lost.**

Producers and consumers scale independently from brokers

Production and consumption rates (e.g. spike or slow consumer issue) have no effect on the broker. **THE STREAM GOES ON.**

Event streams can be enriched in real-time with stream processing

ksqlDB and Kafka Streams **enable event streams to be processed "in-flight"** rather than with a separate batch solution

Message System Benefits

+

Real-time (low latency)

Broad Adoption (familiar technology)

Message System Drawbacks

−

Lacks Common Structure

No Stream Processing

No Persistence After Consumption

Low Fault Tolerance at Scale

Slow Consumers Drag Performance

Complexity and Technical Debt

Message Sys

+

Real-time (low lat

Broad Adoption (

Confluent

+

Real-time

Broad Adoption

+

Schema Registry

Stream Processing

Durable & Persistent

Elastically Scalable

Reliable

n Drawbacks

−

ommon Structure

tream Processing

fter Consumption

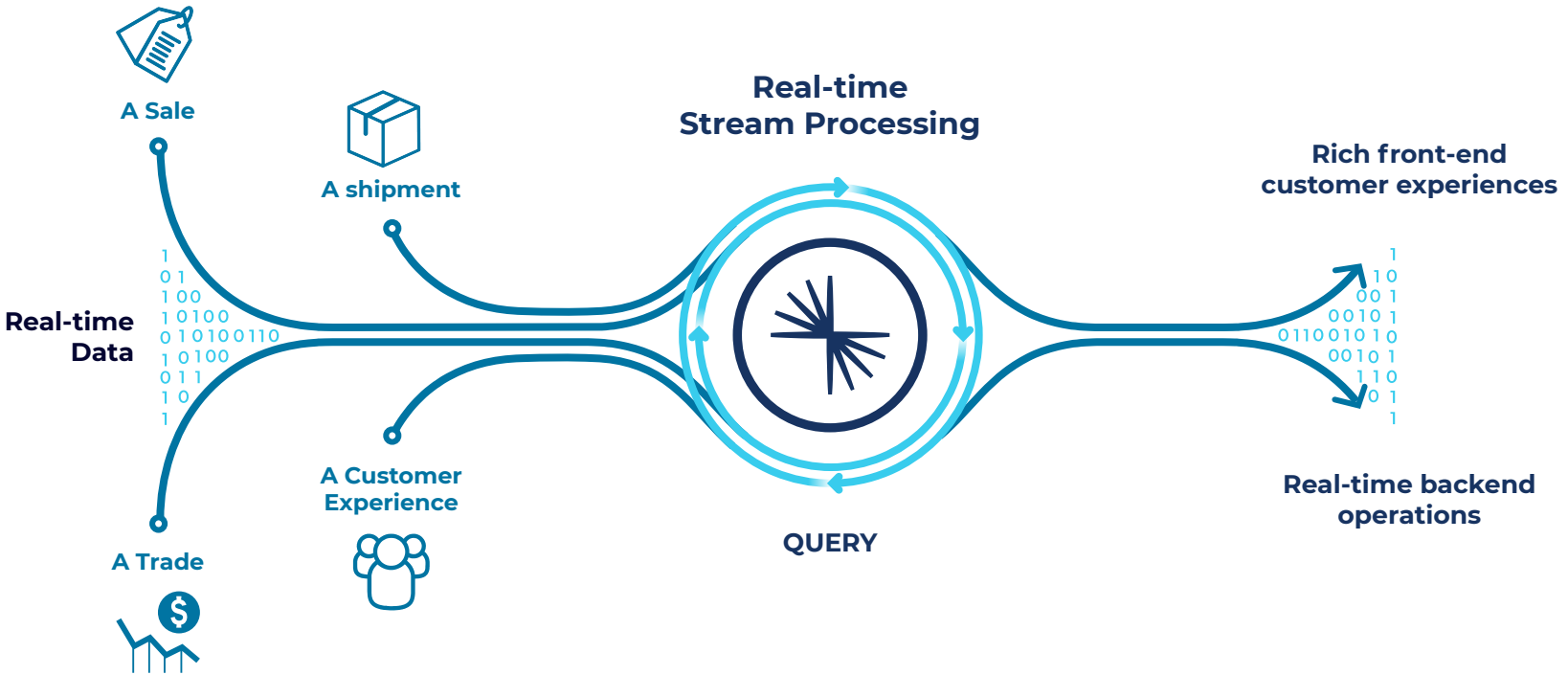
Tolerance at Scale

Drag Performance

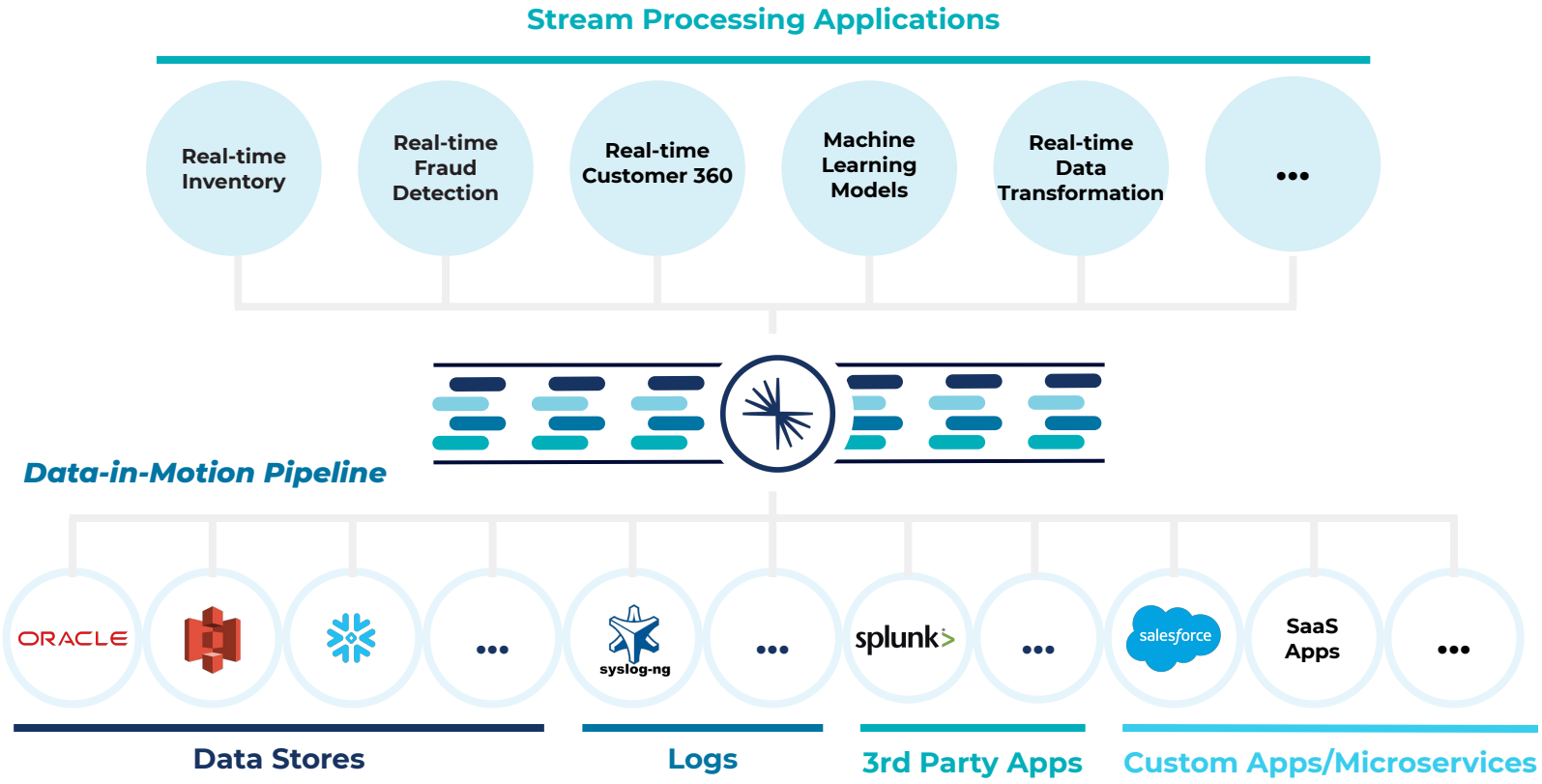
Complexity and Technical Debt

Introduction to Confluent

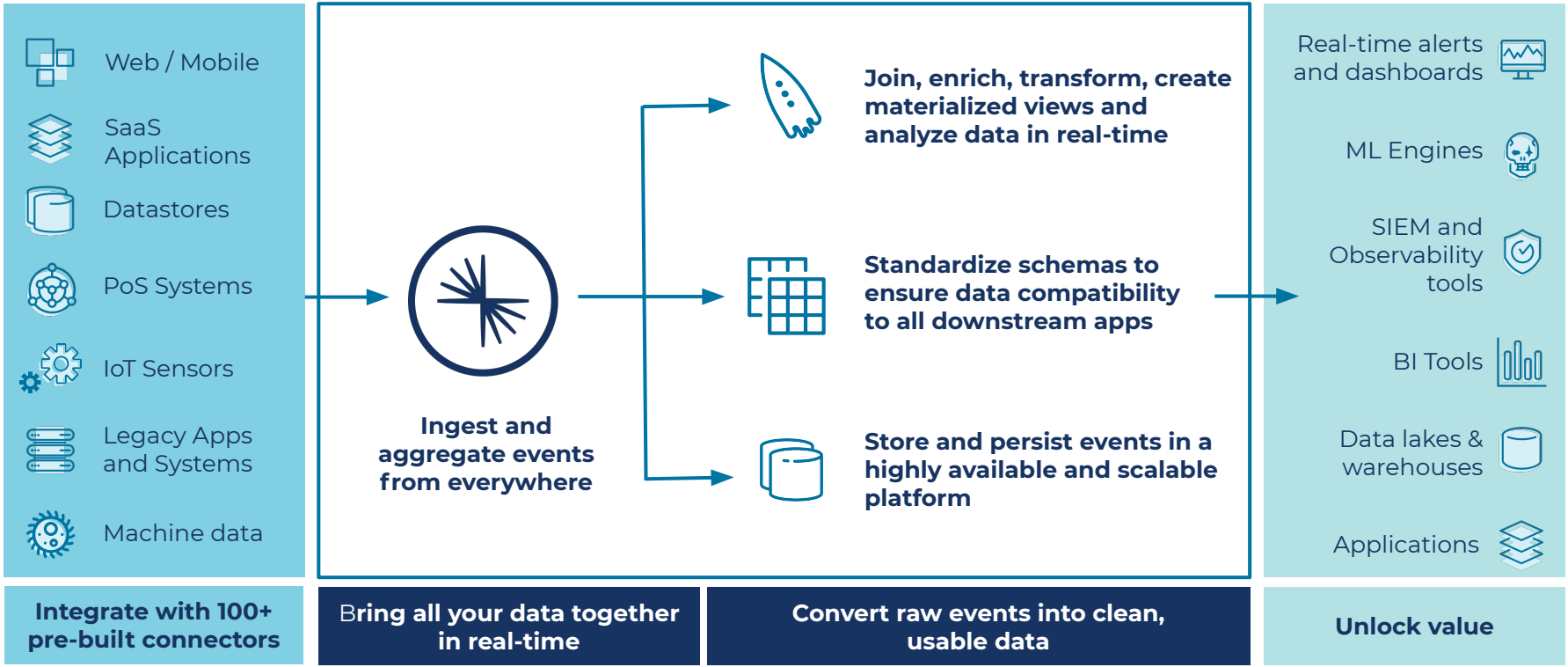
Confluent: A New Paradigm for Data-in-Motion



Implication: Central Nervous System for Enterprise



How Confluent Works



Confluent Modernizes Enterprise Messaging Infrastructure

Build a modern architecture

Enable real-time workloads and applications

Achieve any scale

Future-proof data architecture and achieve high performance and availability at scale

Migrate with ease

Easily augment or migrate apps to Confluent

Reduce TCO and tech debt

Future proof cloud-ready architecture

Build a modern architecture

Confluent offers a way to modernize your data architecture and accelerates your developer velocity

Cloud-native design

Embrace modern DevOps and easily deploy, manage, and scale your infrastructure, whether on your private cloud or a public cloud

Simple integrations

Use best-of-breed solutions across your architecture and easily integrate them with Confluent connectors, reducing your reliance on a single infrastructure vendor

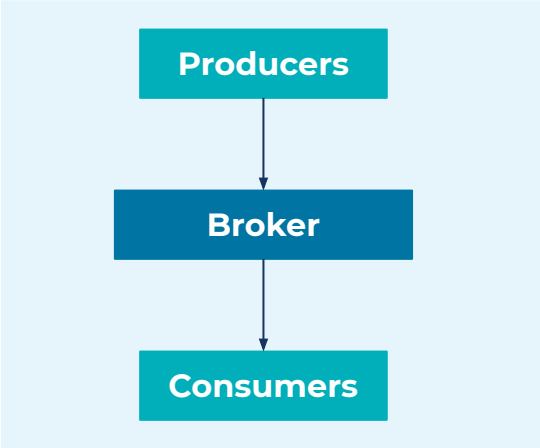
Message replay

Easily replay messages, provide event sourcing and command sourcing for error handling and eliminate complexity for resending messages

Build a modern architecture

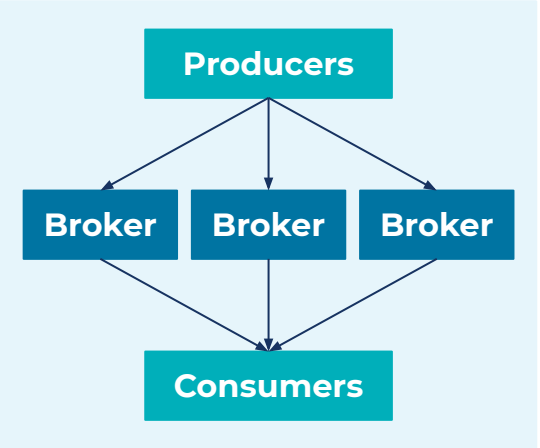
Confluent offers a way to modernize your data architecture and accelerates your developer velocity

Legacy messaging brokers are monolithic



More point of failure, lack of horizontal scalability & higher risk of downtime: poor fit for mission-critical use cases

Kafka supports containerization and multi-DC deployments

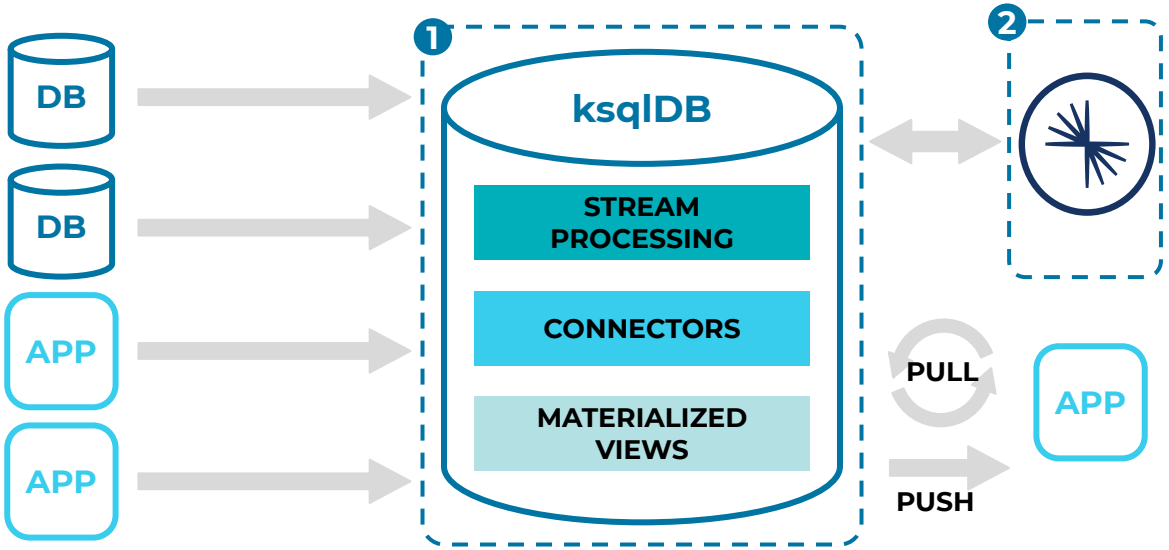


Fault tolerant and horizontally scalable: ensures high availability for mission-critical apps and more cloud-native experience

Build a modern architecture

Confluent offers a way to modernize your data architecture and accelerates your developer velocity

ksqlDB provides everything you need to build a complete real-time application entirely with SQL syntax

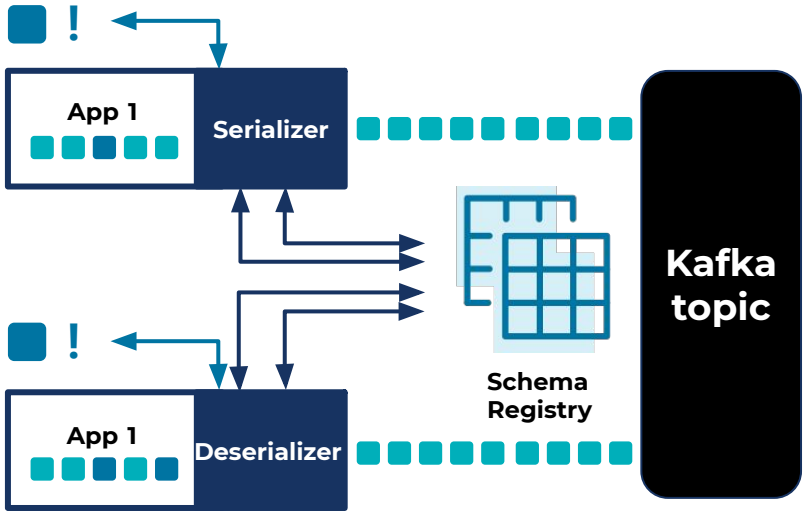


Build a modern architecture

Confluent offers a way to modernize your data architecture and accelerates your developer velocity

Data discovery

- Validate data compatibility and get warnings
- Let developers focus on deploying apps



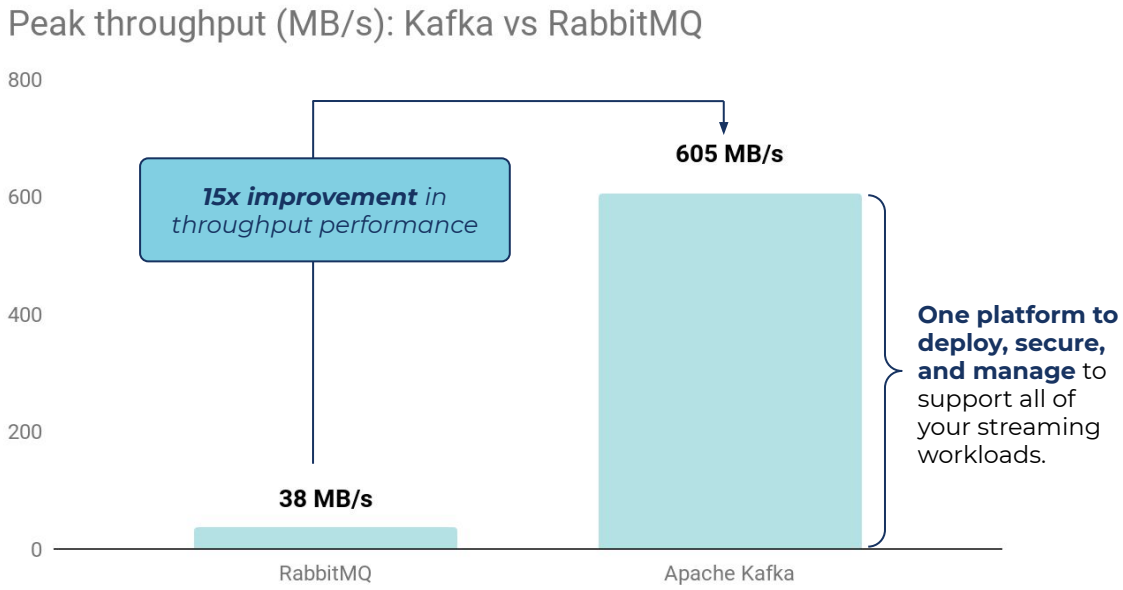
Scale with confidence

- Store a versioned history of all schemas
- Enable evolution of schemas while preserving backwards compatibility for existing consumers

Achieve any scale

Confluent was designed for high performance and high availability at massive scale to support your mission-critical use cases

Confluent can support all of your enterprise's streaming use cases by achieving a dramatically higher throughput



Read more about our internal performance benchmarking

Achieve any scale

Confluent was designed for high performance at massive scale to support your mission-critical use cases

Confluent was designed for high performance at massive scale to support your mission-critical use cases

5ms

Confluent achieves <5ms latency at massive throughput

- Synchronize data across your organization in real-time
- Take action on insights from your data immediately
- Remove data silos by moving from batch to data streaming

Migrate with ease

Confluent provides the tools and services required to effectively migrate from your messaging queue

We offer a robust set of connectors to pull data from your MQs into Confluent...

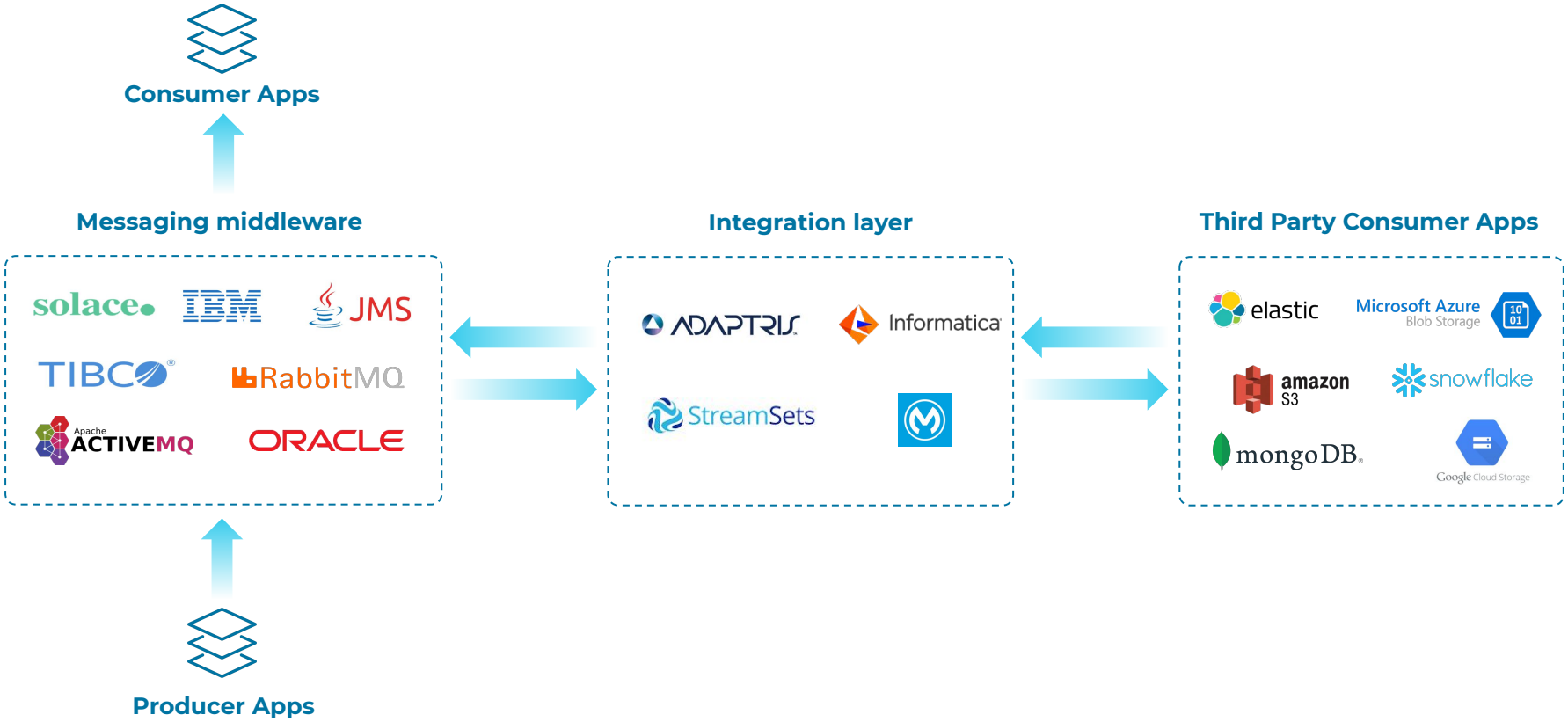
- IBM MQ
- TIBCO
- solace
- JMS
- RabbitMQ
- Apache ACTIVEMQ

...and connectors to push data from Confluent into your modern, cloud-native sinks

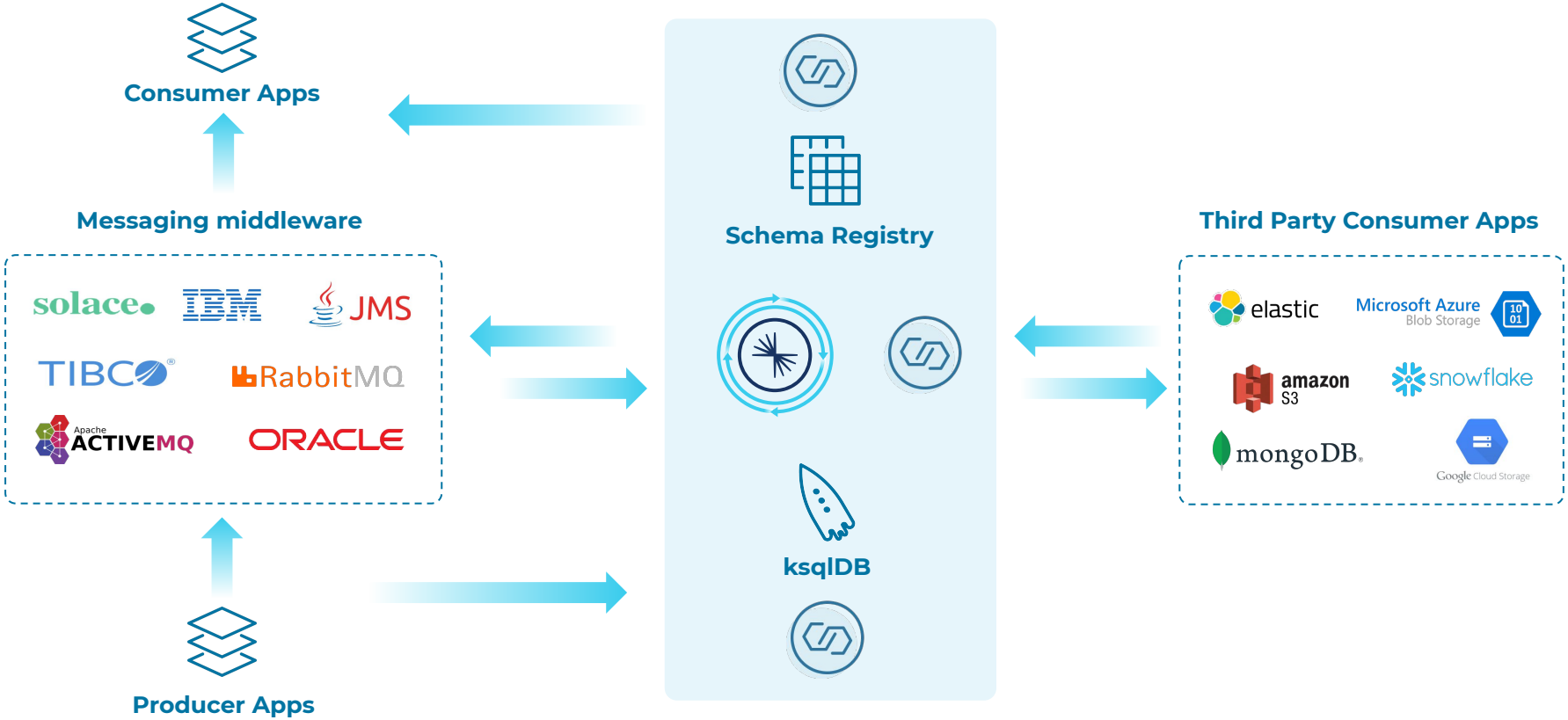
- elastic
- mongoDB.
- Google Cloud Storage
- amazon S3
- Microsoft Azure Blob Storage
- snowflake

How We Do It

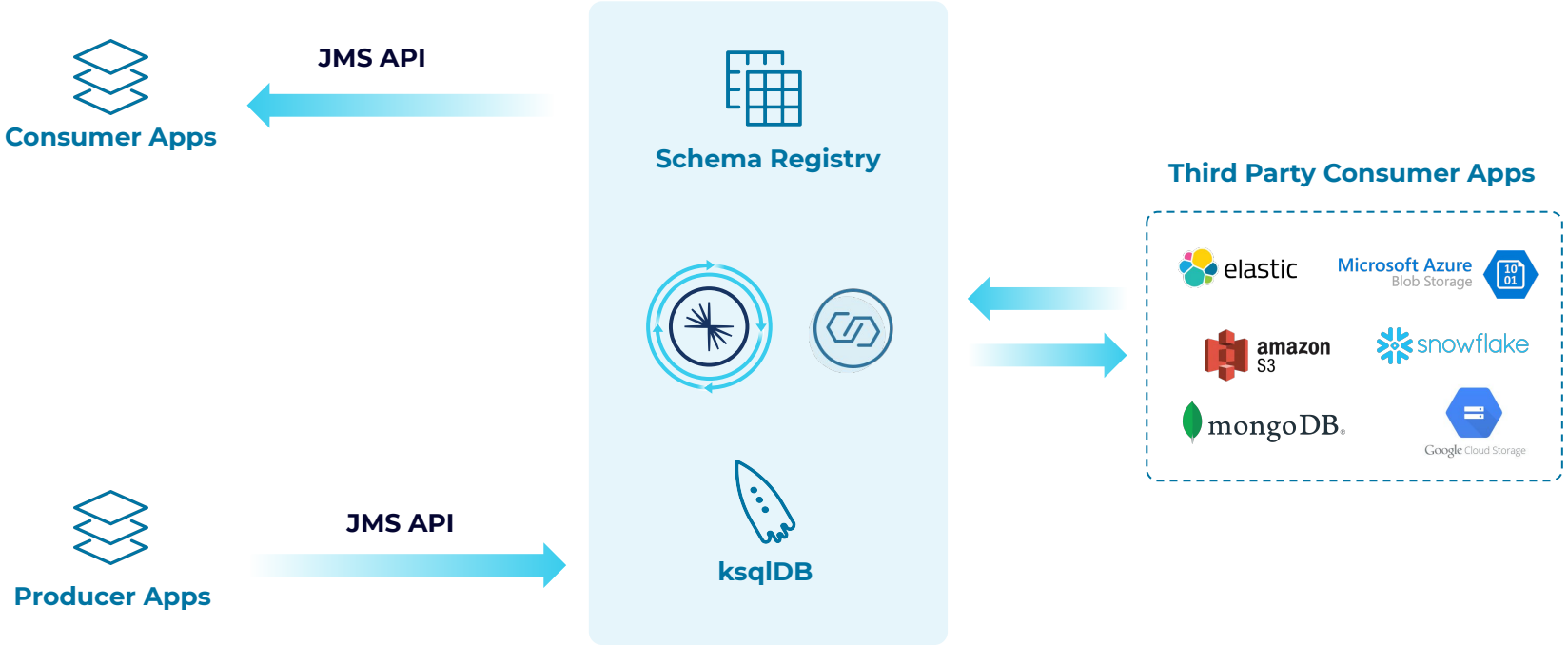
Current approach to Messaging



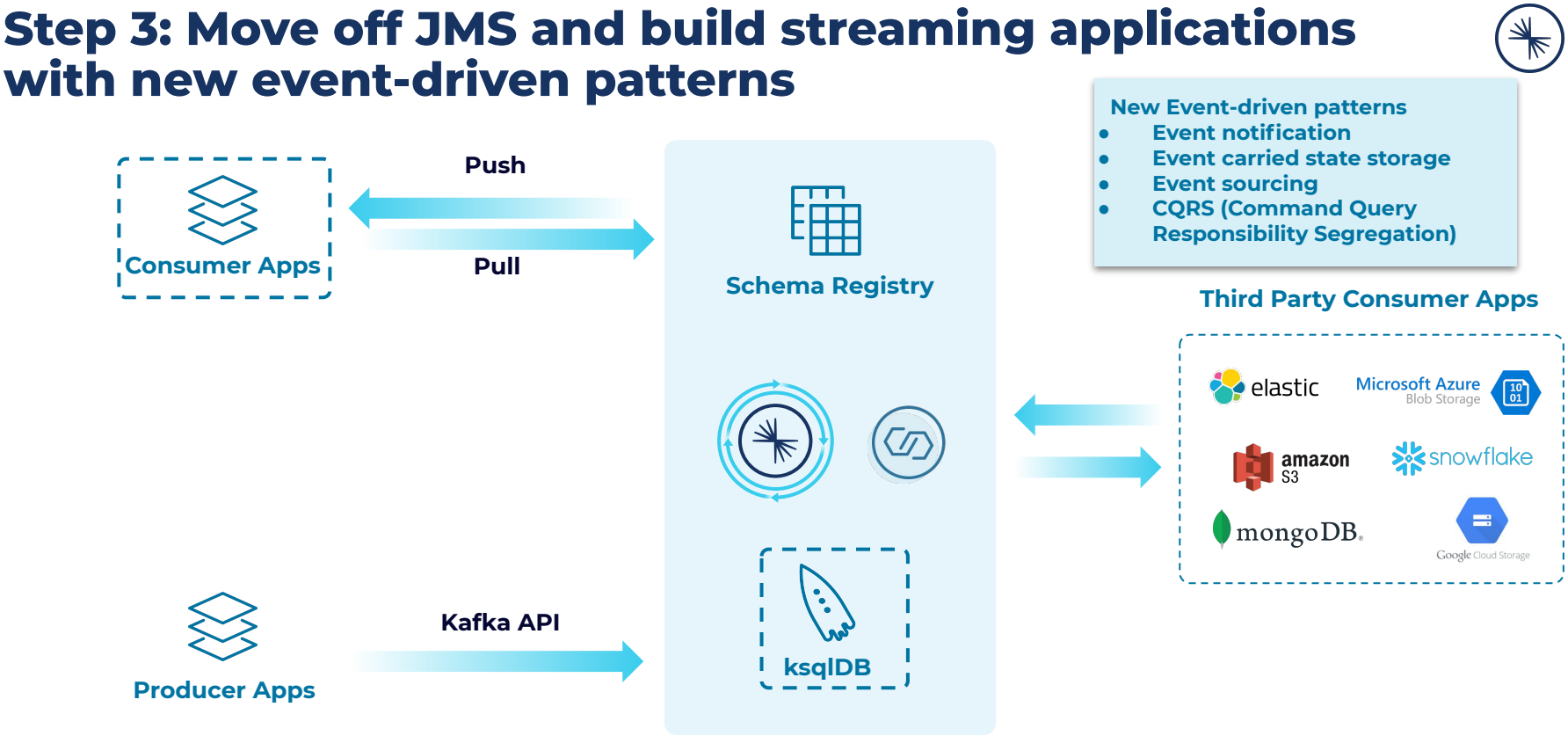
Step 1: Use pre-built connectors to start integrating data from your applications



Step 2: Refactor applications to use JMS APIs and replace messaging middleware over time



Step 3: Move off JMS and build streaming applications with new event-driven patterns



✓ Unify and simplify ✓ Lower TCO ✓ Future-proof data architecture


CONFLUENT