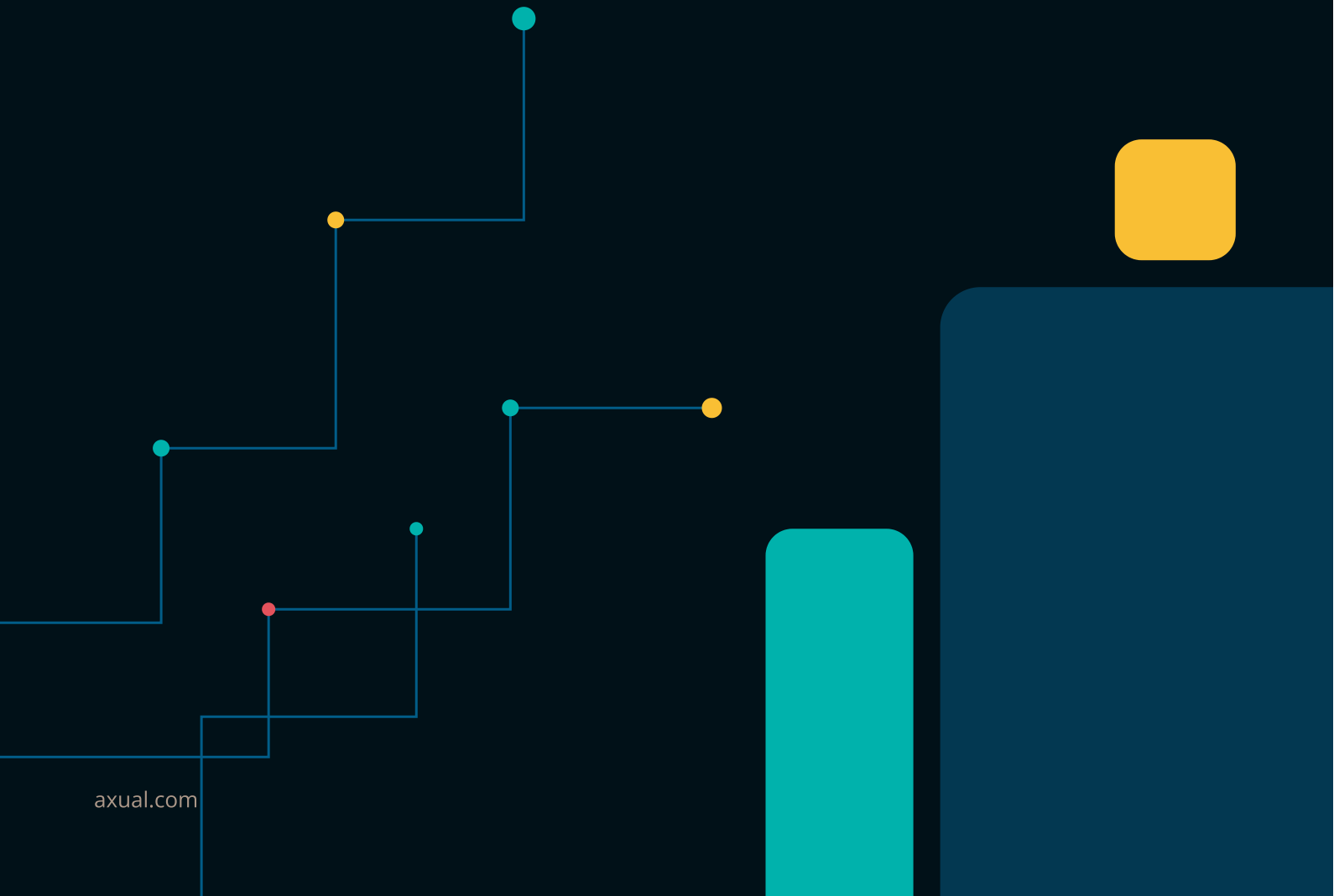
axual

Ebook · Banking

Modern banking with Axual

How banks migrate from batch processing to event streaming architecture.



Executive summary

Event streaming makes banking systems more scalable, reliable and responsive

Event-driven architecture (EDA), data streaming, and Kafka-like systems improve the scalability, reliability, and responsiveness of applications.

Compared to synchronous or batch-based integrations, event-driven architecture reduces downtime, operational overhead, and the impact of system outages.

For banking institutions, event streaming supports:

- Independent communication between systems
- Faster response to changing conditions
- Continuous processing instead of scheduled batch windows
- Persistent and retryable interactions
- Scalable dashboards and data processing

The basics

What is event-driven architecture?

Event-driven architecture is:

01

An integration pattern between systems or components within a distributed architecture

02

Based on asynchronous interactions

03

Built on events that can be persisted and retried

This architecture allows systems to continue operating even when downstream components are unavailable.

Why move now

4 reasons banks must move to real-time event streaming

Traditional batch-based integrations create dependency between systems and delay the movement of information. Here are the four critical reasons financial institutions are making the shift to event-driven architecture.

1 Reduced dependency between systems

Event-driven architecture is a significantly better choice than synchronous or batch-based integrations because it:

- Reduces downtime
- Reduces operational overhead
- Reduces outage impact
- Improves flexibility between distributed systems

2 Improved reliability and durability

Event-driven systems improve reliability because:

- Events can be persisted
- Failed interactions can be retried
- Systems continue functioning asynchronously

This reduces the risk of failed processing when downstream services are unavailable.

3 Scalable data processing

Independent consumers can:

- Build read models optimized for queries
- Create projections for dashboards and reporting
- Process data independently without recalculating everything on demand

This improves scalability for reporting and analytics workloads.

4 Support for dynamic environments

Event-driven architecture supports systems that:

- Continuously change
- Require flexible processing
- Operate across distributed architectures

Use cases

Banking use cases for event streaming

Integration between banking systems

Event-driven architecture can be used as an integration pattern between systems within a distributed architecture.

Benefits

- Reduces dependency between systems
- Reduces operational overhead
- Limits outage impact
- Supports asynchronous communication

Banking use cases

- Integration between core banking systems and downstream services
- Integration between channels and internal systems
- Replacing synchronous or batch-based integrations with event-based communication

Scalable reporting, dashboards and data pipelines

Independent consumers can build read models, query projections and end-to-end data pipelines. Dashboards can read from these projections instead of recalculating everything on demand.

Benefits

- Reduced recalculation of raw data
- Improved dashboard performance
- Scalable analytics processing

Banking use cases

- Reporting workloads across distributed systems
- Dashboard and analytics processing
- Large-scale data processing across multiple internal systems

Reliable processing through persistence and retry

Event-driven systems improve reliability and durability because events can be persisted, failed interactions can be retried, and systems continue functioning asynchronously.

Benefits

- Improved resilience
- Reduced risk of failed interactions
- Better durability of processing

Banking use cases

- Processing interactions even when downstream services are unavailable
- Supporting continuous processing across distributed systems
- Reducing dependency on immediate system availability

Dynamic and distributed banking environments

Event-driven architecture supports systems that continuously change and require flexible processing.

Benefits

- Flexibility across distributed architectures
- Improved responsiveness to changing conditions
- Support for independent consumers and services

Banking use cases

- Distributed banking environments with multiple connected systems
- Systems requiring continuous updates and processing
- Large-scale distributed architectures

Know the trade-offs

Event-driven architecture: challenges and considerations



The open source stack

Open source technologies supporting event streaming

1 Message brokers

- Apache Kafka
- RabbitMQ
- Apache ActiveMQ

2 Stream processing

- Apache Flink
- Apache Spark Streaming
- Apache Samza

3 Event sourcing and CQRS

- Axon Framework
- EventStoreDB

4 Monitoring and observability

- Prometheus
- Grafana
- Elastic Stack (Elasticsearch, Logstash, Kibana)

Benefits of open source technologies

Security and reliability
Broad community review improves visibility into vulnerabilities and robustness.

Transparency
Open source software allows inspection of platform behavior and data handling.

Cost and flexibility
Open source reduces cost and avoids vendor lock-in.

Innovation and community contribution
Community participation accelerates improvements and feature development.

Challenges of open source adoption

Support limitations
Dedicated support may be limited compared to commercial solutions.

Technical debt
Organizations require sufficient processes, talent, and time to manage maintenance and documentation.

The case for real time

Why modern banking requires real-time processing

Modern banking environments require systems that are more responsive, scalable, and continuously available. Traditional batch-based integrations create delays between systems, increase operational dependency, and limit responsiveness in distributed environments.

Financial institutions increasingly require

- Real-time movement of information between systems
- Continuous processing instead of scheduled batch windows
- More scalable and responsive architectures
- Faster response to changing conditions
- More resilient distributed systems
- Flexible communication between independent services

Event-driven architecture and event streaming provide an alternative to synchronous or batch-based integrations by enabling asynchronous communication, persistent and retryable interactions, and scalable distributed processing.

With Axual

Kafka-based event streaming platforms such as Axual help simplify this transition by supporting:

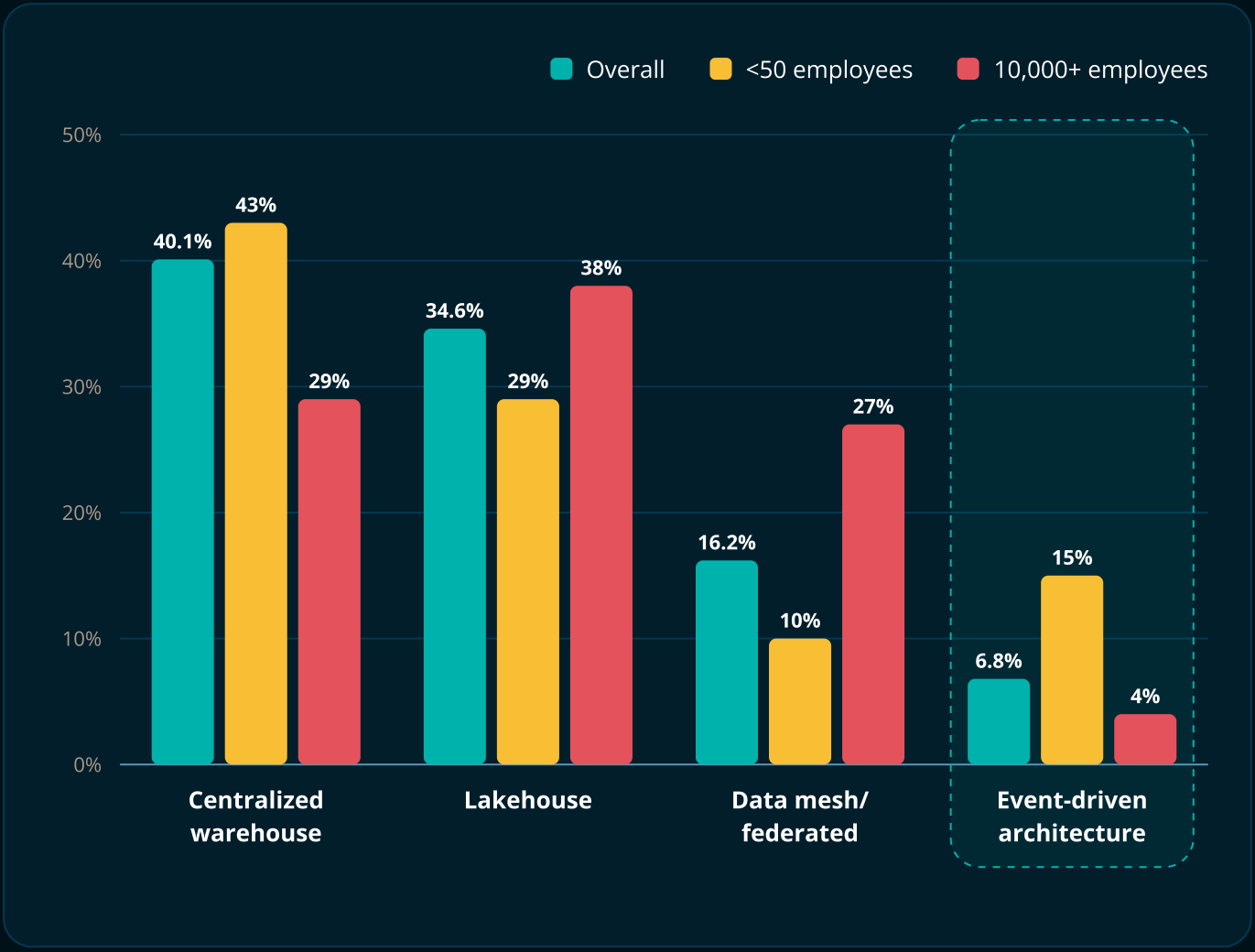
- Integration between distributed systems
- Independent consumers and services
- Persistent event handling
- Scalable data pipelines and dashboards
- Governance and operational visibility
- More flexible and resilient architectures



Market context

Architectural trends in data engineering

According to the 2026 Practical Data Community State of Data Engineering report, organizations continue to favor centralized architectural approaches, though preferences vary by organization size.



The results show that while centralized architectures remain dominant overall, larger enterprises are increasingly exploring federated and distributed architectural models. **Event-driven architecture remains an emerging architectural approach**, particularly within organizations modernizing distributed systems and data integration.

Source: 2026 Practical Data Community State of Data Engineering

Adoption of event-driven architecture in modern banking

Financial institutions are increasingly adopting event-driven architecture and Kafka-based event streaming platforms to modernize integration, improve resilience, and support scalable distributed architectures.

Compared to synchronous or batch-based integrations, event streaming supports:

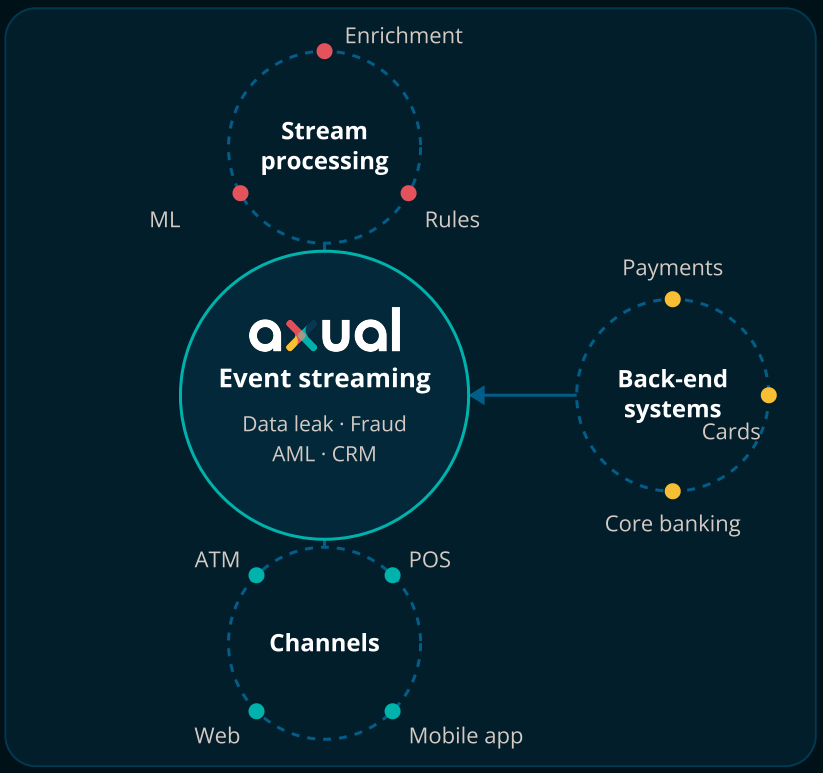
- Continuous processing instead of scheduled batch windows
- More scalable and responsive systems
- Reduced operational overhead
- Independent communication between systems
- Persistent and retryable interactions
- Scalable dashboards and analytics processing
- Flexible distributed architectures

Banks are adopting event streaming to support:

- Integration between distributed banking systems
- Reliable asynchronous processing
- Scalable reporting and dashboards
- End-to-end data pipelines
- Dynamic and continuously changing environments

Kafka-based event streaming platforms such as Axual support this transition by enabling:

- Independent consumers and services
- Asynchronous communication
- Persistent event handling
- Scalable distributed architectures
- Governance and operational visibility



Proof

Evidence from banking implementations

ptsb

10s → <1s
payment latency

Results

- Payment latency reduced from 10 seconds to less than 1 second
- 100% SEPA transaction compliance
- Time to production achieved in 4 months
- 99.99% platform uptime

Governance capabilities

- Governance-first architecture
- Centralized governance layer
- Self-service provisioning

asn bank

50+
development teams onboarded

Results

- 50+ internal development teams onboarded
- One centralized Kafka platform replacing fragmented infrastructure
- Multiple availability zones providing automatic failover and recovery

Governance capabilities

- Role-based access control
- Approval workflows
- Schema and data governance
- Data lineage tracking
- Audit capabilities

Rabobank

€500,000
saved annually in maintenance

Results

- 500 million messages processed daily
- 600+ production data streams
- €500,000 annual savings in maintenance costs
- 300+ DevOps teams enabled with self-service
- 100% GDPR compliance maintained
- Zero unplanned downtime

Governance capabilities

- Mutual SSL integration
- Role-based access control
- Data ownership models
- Audit of every change
- Approval workflows
- Team-specific namespaces

Summary

Key takeaways

01

Why batch-based integrations create operational overhead and dependency between systems

02

Why event-driven architecture is becoming more relevant for modern banking environments

03

How asynchronous processing supports more resilient distributed architectures

04

How event streaming improves scalability, reliability, and responsiveness

05

How independent consumers and scalable data pipelines support reporting and analytics workloads

06

How governance capabilities support more controlled and scalable adoption of event streaming

07

Why financial institutions are adopting Kafka-based event streaming platforms such as Axual

08

How banks such as Rabobank, PTSB, and ASN Bank are using event streaming to modernize infrastructure and operations

Conclusion

Built for distributed, always-on banking

Financial institutions are operating in increasingly distributed and dynamic environments that require more scalable, reliable, and responsive systems.

Compared to synchronous or batch-based integrations, event-driven architecture provides a more flexible approach to system integration and data processing. Through persistent and retryable events, independent consumers, and asynchronous communication, event streaming supports more resilient and scalable banking architectures.

Banks also require more operational visibility, more flexibility between systems, more scalable reporting and analytics, and more reliable processing across distributed environments.

Kafka-based event streaming platforms such as Axual support this transition by enabling more scalable distributed architectures, more resilient integration patterns, and more controlled governance capabilities for modern banking environments.

See it in practice

From batch processing to real-time event streaming

See how we helped a major banking organization make the move from batch processing to real-time event streaming.

Book a demo →

axual.com

axual