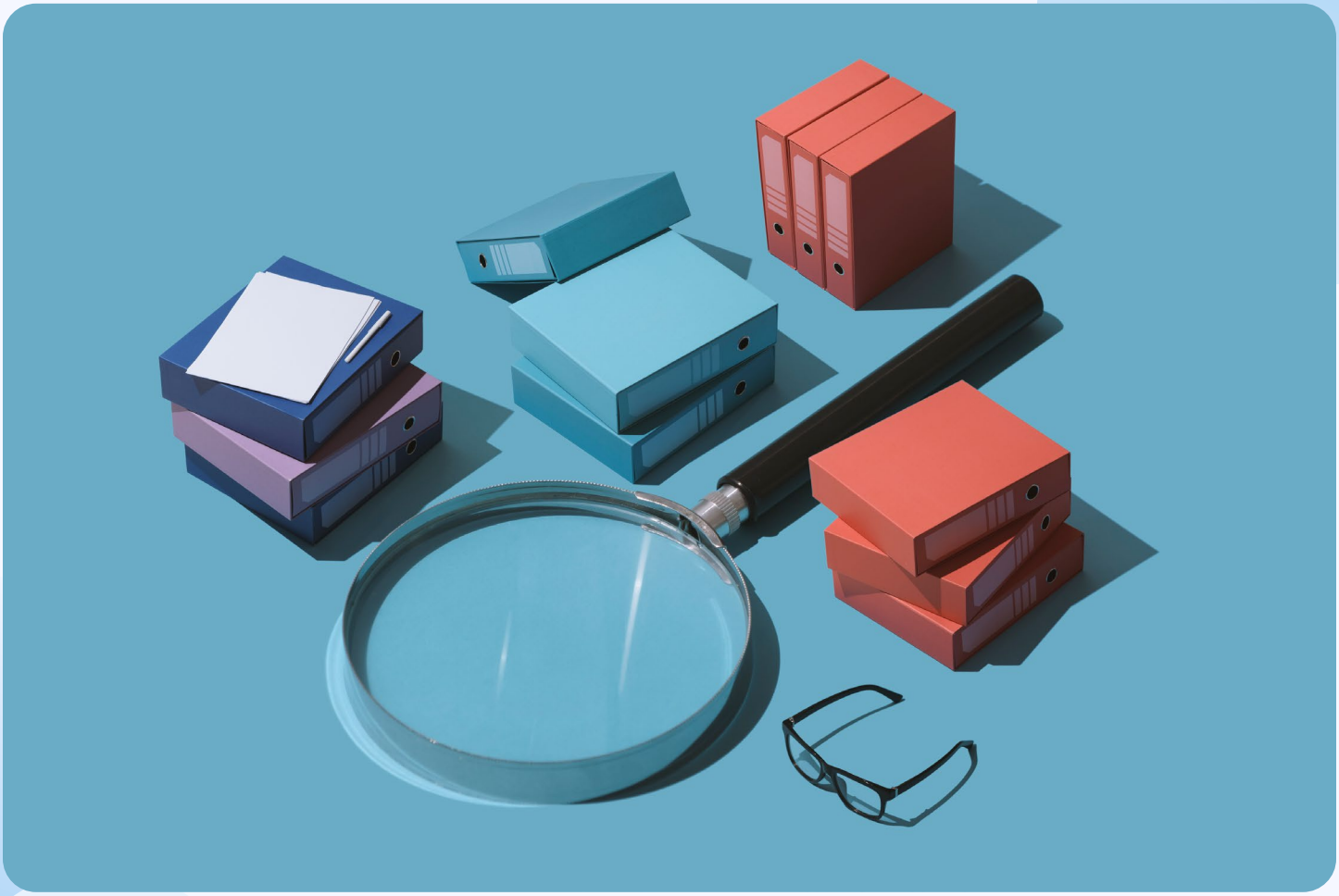
The Top 6 Reasons Kafka Projects Fail

(and How to Overcome Them)

John Santaferraro, CEO and Head Research Analyst of Ferraro Consulting



CONFLUENT | FERRARO Consulting

confluent.io

Table of Contents

Click the page number on any page if you'd like to return to the Table of Contents.

Introduction

THE DIGITAL DELUGE of the last decade has elevated streaming data to a place of prominence in the IT landscape. According to the Apache Foundation, Apache Kafka® is used by thousands of companies and over 80% of the Fortune 100, including companies like Box, Goldman Sachs, Target, Cisco, Intuit, and more. It has become the de facto choice for data in motion.

Kafka is an open source distributed messaging system designed to handle large-scale data streams using events. With more than 1,000 use cases and counting, Kafka is proving to be far more than a traditional messaging platform; it has become the central nervous system of countless IT ecosystems. Organizations deploy Kafka for their critical, real-time transactions, including payment processing, financial transaction monitoring, and fraud and risk detection. Kafka provides an extensive set of tools for pub-sub messaging, stream processing, system connectivity, and ETL tooling.

Although Kafka possesses powerful and transformative capabilities, it does require dedicated personnel, resources, and infrastructure to set up, configure, monitor, and manage. Multiple clusters with high streaming data volume adds additional complexity, especially as you onboard new business processes, applications, and use cases.

Kafka is a distributed system, and like most distributed systems, operating it requires careful consideration of networking, resource allocation, faults, and recovery. While it is relatively easy to get started in using Kafka, building a reliable and trustworthy Kafka service can take many weeks or even months. And this isn't unique to Kafka—introducing and self-managing any essential service into your organization takes significant time, effort, and money to do well. The unavoidable fact is that every minute spent building out foundational components and performing management tasks takes time away from building rich customer experiences and improving your business efficiency with data-driven operations.

There are six common reasons why Kafka projects fail:

- Lack of expertise and resources
- Difficulty moving from development to production
- Unpredictable outages and downtime
- Difficulty securing streaming data
- Lack of governance
- Difficulty scaling

This report explores each of these six reasons.

Kafka Risk 1

Lack of Expertise and Resources

DEVELOPING, DEPLOYING, MONITORING, scaling, and implementing Kafka solutions requires a high degree of expertise. Installation can be straightforward, but architecting these event-driven data pipelines and deploying them for enterprise use often requires outside resources, such as consultants who are familiar with modern streaming architectures. Insufficient understanding of Kafka's concepts, architecture, and best practices can lead to poor implementation decisions. The overall lack of resources and expertise falls into three areas: installation, development, and operations.

Installation

Despite straightforward installation, some failures occur because hardware and network configurations are underprovisioned. But overprovisioning carries a risk of overspending for underutilized resources. There are many decisions to be made regarding a physical machine or a cloud provider, including all of the intricacies of each option. The network needs to be configured properly for streaming data, especially for potential unforeseen peaks. In addition, hardware and network security often operate on separate systems.

Development

For developers new to working with distributed systems, there are many new complex concepts to understand, including partitioning, replication, fault tolerance, and consistency. In addition, Kafka has its own ecosystem that must be learned, with concepts such as brokers, clusters, partitions, topics, logs, producers, and consumers. Once developers understand the architecture, they will need to learn how to optimize performance by understanding the trade-offs in order to make informed decisions based on factors like message throughput, latency, disk space, and network bandwidth.

Kafka itself is also an open source project with many powerful tools, but it isn't a complete data streaming platform. Connectors, security and governance features, geo-replication tools, and other functionalities need to be integrated from third parties or custom-built in-house. For example, metadata catalogs, role-based access control, and monitoring tools are not provided as part of the core Kafka project, and must be developed and integrated on your own. The development of these additional components can become increasingly time-consuming and costly.

Operations

Once in production, ongoing operations and maintenance are essential for a Kafka project's success. Establishing effective monitoring practices is necessary to capture important metrics, such as message throughput, latency, and resource utilization. Administrators struggle to set up alerts for proactive identification and resolution of issues to ensure the smooth functioning of Kafka clusters. While there are third-party tools built for monitoring Kafka operations, the challenge comes in trying to unify the monitoring of hardware, network, storage, and compute. Without a unified view across the entire system, it is difficult to detect faults, identify the root cause, and understand the impact of potential failures. Plus, regular maintenance tasks will be necessary such as log retention management, cleaning up old data, and optimizing resource utilization.

Kafka Risk 2

Difficulty Moving from Development to Production

THE PROCESS OF moving streaming data projects from development or proof of concept to production is always more challenging than anticipated, especially for companies new to Kafka. Common mistakes include development and testing without the necessary production components to properly simulate an enterprise environment. Organizations attempting to manually build or piece together solutions with configuration management, high availability, security and access control, performance testing, and monitoring are at high risk for extensive delays and possible failure.

Configuration Management

Configuring Kafka for production involves fine-tuning various Kafka parameters such as buffer sizes, replication factors, retention policies, and security settings. Ensuring that the configuration is optimized for performance, scalability, and security can be challenging, as it requires a deep understanding of Kafka's internals and the specific requirements of your production environment compared to your test environment.

High Availability and Fault Tolerance

Kafka deployments in production should be designed for high availability and fault tolerance. This involves configuring replication to ensure data is replicated across multiple brokers, and implementing proper disaster recovery mechanisms. Teams need to implement strategies for managing broker failures, handling leader elections, and maintaining data consistency across replicas. However, third-party data replication tools for Kafka such as Apache Kafka MirrorMaker 2.0 require provisioning, tuning, and managing extra software components or distributed systems to copy messages from one cluster to another.

Security and Access Control

In a production environment, open source Kafka needs to be secured to protect against unauthorized access and data breaches. However, configuring secure communication channels, enabling authentication and authorization mechanisms, such as access control lists (ACLs), and managing access control policies can be complex—especially on top of managing the underlying Kafka infrastructure.

Performance Testing and Optimization

The biggest challenge of testing comes in setting up an environment that adequately mimics the ultimate production environment. It is complicated to create scalability and performance testing to ensure your Kafka setup can handle the anticipated data volumes and load in a production environment. This may require testing the scalability of Kafka clusters and optimizing configurations to achieve ideal performance. With additional data coming into a production environment, it is important to load balance. Achieving an even distribution of message traffic across a cluster to avoid overloading specific brokers requires careful monitoring and dynamic partition reassignment. It is equally challenging to rebalance partitions and redistribute workloads without impacting ongoing data processing and delivery.

Monitoring

Monitoring the health and performance of your Kafka cluster in production is vital for identifying and resolving issues proactively. While observability itself is not difficult, monitoring Kafka clusters to visualize Kafka operations, alongside all the infrastructure, is difficult. Typically, this requires leveraging a third-party tool such as Prometheus or Grafana to collect and help visualize relevant metrics and configuring alerts based on thresholds or anomalies. Without this cross-platform view, it is impossible to do root cause analysis, impact analysis, or predictive maintenance.

Kafka Risk 3

Unpredictable Outages and Downtime

DUE TO THE distributed nature of Kafka, many projects are susceptible to various risks that can result in unpredictable outages and downtimes. There are multiple challenges with running Apache Kafka on your own, requiring hundreds of man hours of development time, just trying to stitch together reliable infrastructure, network, and software so they all work together. In addition, the amount of time spent on break and fix issues consumes valuable cycles that could be devoted to delivering a project on time and on budget.

Data Integrity and Replication Challenges

Because distributed computing requires frequent replication of data, the chance for errors is multiplied. With Kafka, data integrity and replication across multiple clusters is crucial for fault tolerance and reliability. If data replication mechanisms fail or encounter inconsistencies, it can lead to data loss, inconsistencies across partitions, and potential service disruption.

Complexity of Infrastructure, Network, and Software

The complexity of manually deploying Kafka with high availability and resiliency is compounded by the matrix of infrastructure, network, and software required for on-premises deployments. Development requires expertise in compute, storage, and networking, along with Kafka. Without deep collaboration among the different technology experts, Kafka deployments will remain unstable. Ensuring proper load balancing, network partition tolerance, and fault tolerance requires robust network configurations, hardware setups, and fault-tolerant architectures.

Improper Tuning of Timeouts

Timeouts are typically used in communication between Kafka clients and brokers or between different components in the Kafka ecosystem. If timeouts are set too low, it can result in premature termination of operations, leading to message loss or disruptions in data processing. On the other hand, setting timeouts too high can lead to delays in detecting and recovering from failures, potentially causing prolonged downtime. Configuring appropriate timeouts and handling timeout-related errors effectively is crucial to maintain the stability and reliability of Kafka projects.

Configuration Errors

Incorrect or misconfigured settings within Kafka clusters, such as replication factor, partition allocation, or resource allocation, can lead to performance degradation or even complete failures. Improper configuration changes without adequate testing can introduce instability and downtime.

Kafka Risk 4

Difficulty Securing Streaming Data

EVEN THOUGH KAFKA continues to improve security options for users, ensuring data privacy and protection for streaming data can be challenging for most organizations. Unfortunately, not all security experts understand streaming data and not all streaming data experts understand security.

The complexity of deploying enterprise-grade security requires both Kafka and security expertise, a combination that is rare and in high demand.

To better understand the challenges of streaming data security, we must first understand the full set of security requirements. Data security should protect digital assets from unauthorized access or loss of any kind. Specifically, it should include a full set of security functions including authentication, access control, encryption, key management, configuration assurance, monitoring, and auditing. The risks involved in properly setting up security include the following:

Inadequate Authentication and Access Control

Failing to implement proper authentication mechanisms can result in unauthorized access to Kafka topics or brokers. This can lead to unauthorized data consumption, tampering, or leakage.

Improper Key Management

Encryption in Kafka often relies on key-based algorithms. Inadequate key management practices, such as weak key generation, improper storage, or inadequate rotation, can weaken the overall security posture. Compromised keys can result in unauthorized access or decryption of sensitive data.

Misconfiguration of Security Features

Misconfigurations, such as incorrect SSL/TLS certificate setup, incorrect authentication mechanisms, or improper access control lists (ACLs), can inadvertently introduce vulnerabilities. Many organizations struggle to set up the rigorous reviews and validations necessary to ensure compliance of Kafka security configurations.

Insufficient Encryption

Lack of encryption measures, such as SSL/TLS encryption, can expose streaming data to eavesdropping and interception by malicious entities. Ensuring end-to-end encryption in a distributed environment, where data flows across multiple components and networks, is especially difficult.

Lack of Monitoring and Auditing

Insufficient monitoring and auditing of security-related events in Kafka can make it challenging to detect and respond to potential security breaches. Without proper visibility into access patterns, authentication failures, or suspicious activities, it becomes difficult to identify and mitigate security risks promptly.

Kafka Risk 5

Lack of Governance

UNGOVERNED KAFKA ENVIRONMENTS make it difficult to find, understand, and trust streaming data. It is not uncommon for developers to create new topics without documenting the source of the data, the data owners, proper definitions for data transformations, and use cases, leaving data consumers baffled. Even if some topics are well governed, the mixture with ungoverned data brings risk.

As the number of topics grows into hundreds and even thousands, the risks increase exponentially due to added dependencies and resource conflicts. The lack of governance leads to project failure when issues with data quality, consistency, and explainability abound.

While the tools to build and maintain long-term, compatible event streams do exist, the tools to safely and effectively share these streams across teams for more widespread use do not. This missing piece is extremely problematic for event-driven microservices built by small, disparate teams. This issue is compounded with scaling, causing confusion among producers and consumers of data.

While most Kafka project leaders set out to govern streaming data, many fall short of true governance, which includes all of the following:

Data Consistency

When organizations fail to implement data cleansing across the entire data ecosystem, it results in inconsistent data. Inconsistency breeds inaccurate analysis and insight.

Data Quality

The number one governance challenge is achieving the high-speed data cleansing necessary for extracting value from streams. There is no built-in mechanism to enforce data validation rules, data formats, or data schema evolution.

Data Lineage

Without a complete understanding of where data originates and how data has been manipulated, it is impossible to derive meaningful intelligence.

Data Policies and Standards

Consistent and appropriate handling of data is impossible without well-defined guidelines and rules for data management, including data classification, access controls, data retention, and privacy regulations.

Data Stewardship

When organizations fail to identify data owners, it is impossible to implement the policies and standards necessary to operate a well-governed data practice.

Data Catalog

Missing or incomplete data catalogs make it difficult to find and understand data in the context of the greater organization. Kafka does not have a centralized repository or catalog to track and manage metadata information such as topic schemas or message formats.

Kafka Risk 6

Difficulty Scaling

BECAUSE KAFKA COVERS a broad range of use cases, from messaging all the way to data integration and data pipelines, opportunities abound to scale and expand use. However, scaling Kafka often requires manual intervention, resulting in the following challenges:

Globalization

When scaling to new regions, balancing workloads and synchronizing data across replicated clusters presents its own challenges, especially when considering different data privacy and security requirements in different regions. Furthermore, managing data replication within Kafka means ensuring consistency across replicas, synchronizing data updates, and handling replica failures.

Performance Bottlenecks

As the data load increases, resource conflicts arise, and Kafka clusters may face performance bottlenecks. These bottlenecks can occur at various levels, such as network bandwidth, disk I/O, or CPU utilization. Identifying and mitigating these bottlenecks while maintaining optimal throughput and latency can be challenging. The balance of infrastructure and performance upgrades is difficult.

Long-Running Topics

When there is an increase in the number of topics running over an extended period of time, it impacts both storage and data retention. Storage requires compression and other optimization strategies. In addition, when consumers lag behind producers, data gets backed up and can easily be lost or duplicated.

Maintenance

Maintenance at scale requires understanding of cluster health, disk utilization, and network latency in complex Kafka environments, a skillset that is difficult to find. Regular maintenance tasks like backups, upgrades, configuration adjustments, and resource allocation adjustments require visibility, monitoring, and analysis solutions to ensure stability.

Orchestration

As data streams proliferate and Kafka becomes the central nervous system of the enterprise, the complexity of managing competing workloads outgrows human capabilities, especially when different departments in the business are competing for resources. Orchestration requires a rich set of centralized metadata to guide the automation of resource allocation, conflict resolution, and error correction.

Utilization Optimization

As Kafka streams multiply, utilization remains a mystery to many organizations. The lack of centralization in on-premises deployments leads to overprovisioning, underprovisioning, and data loss. Optimization for either cost or performance can be difficult.

Repairs

Many organizations expanding the use of Kafka spend an inordinate amount of time on break and fix issues. Inadequate standardization and inconsistent configurations compound the matter further.

The Undeniable Benefit

Confluent vs. Apache Kafka

USING A FULLY managed platform to address open source complexity is particularly useful for Kafka deployments. There are several areas where Confluent has built technology around Kafka to reduce risk, speed time to streaming excellence, and create value for customers. Confluent provides a truly cloud-native experience, completing Kafka with a holistic set of enterprise-grade features to unleash developer productivity, operate efficiently at scale, and meet all of your architectural requirements before moving to production.

Here's a quick rundown on the specifics of ensuring success upfront for event-driven data streaming projects.

OVERCOMING RISK 1

Elimination of Resource and Expertise Challenges

Confluent Cloud eradicates time-consuming cluster provisioning and configuration, which are common roadblocks for teams moving beyond a Kafka proof of concept. Instead of putting internal resources to work maintaining Kafka clusters, Confluent's fully managed, cloud-native platform removes any operational overhead burden. This means you can get started with your Apache Kafka deployment in minutes. Confluent draws its experience from more than 1 million cumulative hours of Kafka expertise and operating more than 30,000+ clusters across thousands of organizations.

OVERCOMING RISK 2

Quickly Move from Development to Production

Confluent Cloud handles all operational aspects of Kafka for you, including provisioning, scaling, monitoring, and maintenance, with features like connectors, stream processing, governance, and more out of the box. Developers also don't need to worry about production necessities like disaster recovery because Confluent Cloud ensures high availability and resilience by handling the replication and data durability aspects of Kafka. Resources can also be quickly spun up and turned down, giving teams access to right-sized testing environments and streamlining the move from development to production.

OVERCOMING RISK 3

Unpredictable Outages and Downtime

Here, the SLA that Confluent Cloud brings—99.99% uptime—relieves a lot of the worry and unpredictability of a self-managed Kafka deployment. Built-in capabilities like multi-availability zone clusters and automated Kafka patches also help alleviate downtime concerns. Better yet, Confluent Cloud supports multi-region deployments, allowing you to replicate your Kafka data across multi-availability zones or cloud regions. So in an event of an outage in one region, Confluent Cloud will automatically fail over to another region, minimizing your downtime.

OVERCOMING RISK 4

Securing Your Data Streams

Confluent Cloud offers robust security measures to ensure the confidentiality, integrity, and availability of your Kafka data. It ensures encryption at rest and in transit, safeguarding your data from unauthorized access through security features such as role-based access control and audit logs. It complies with industry standards and certifications, and offers fine-grained security monitoring and threat detection.

OVERCOMING RISK 5

Acceleration and Assurance of Governance

Stream Governance is the industry's only fully managed data governance suite for Apache Kafka. With three integrated components—stream quality, stream catalog, and stream lineage—development teams can bring the power of real-time data streams to your business safely and confidently.

OVERCOMING RISK 6

Scaling Without Limits

Because Confluent Cloud is fully managed, you no longer need to worry about the operational overhead (e.g., maintenance, Kafka upgrades, and patches) that comes with scaling your Kafka deployment. Confluent Cloud also allows you to easily scale your Kafka resources up and down based on your requirements. You can adjust the capacity of your clusters to handle increased data volumes, higher throughput, or spikes in traffic.

ORGANIZATIONS THAT USE Confluent have launched their streaming projects into production over six months faster than with Apache Kafka on their own, all while drastically reducing their day-to-day operational burdens, risks, and costs by up to 60%. Confluent has reinvented Kafka so you can scale Kafka across your organization without the need to scale your teams.

Interested in learning more?

Contact us today to see how Confluent can help ensure your success with Kafka and data streaming.

Or is your team ready to try Confluent?

Get Started for Free Today