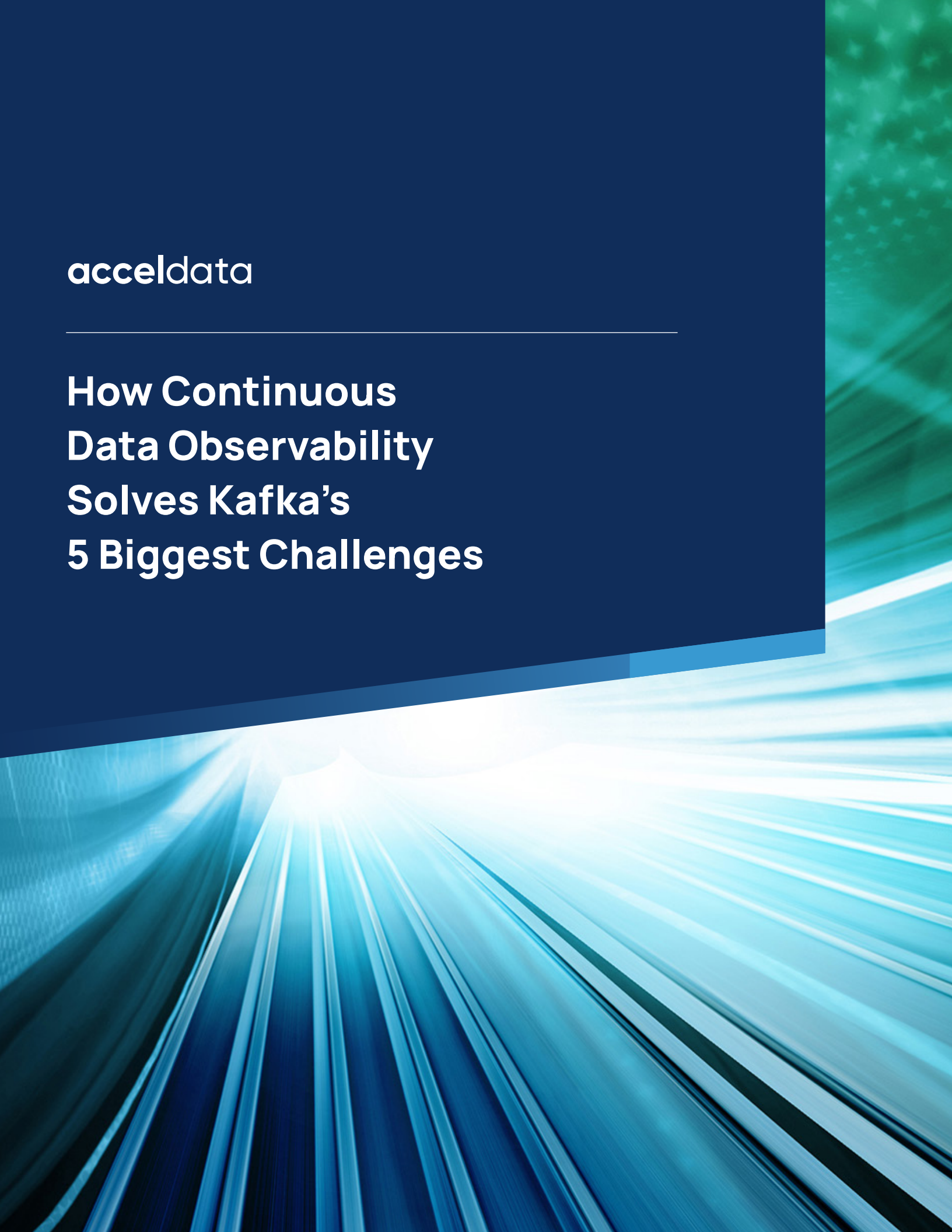
acceldata

How Continuous Data Observability Solves Kafka's 5 Biggest Challenges



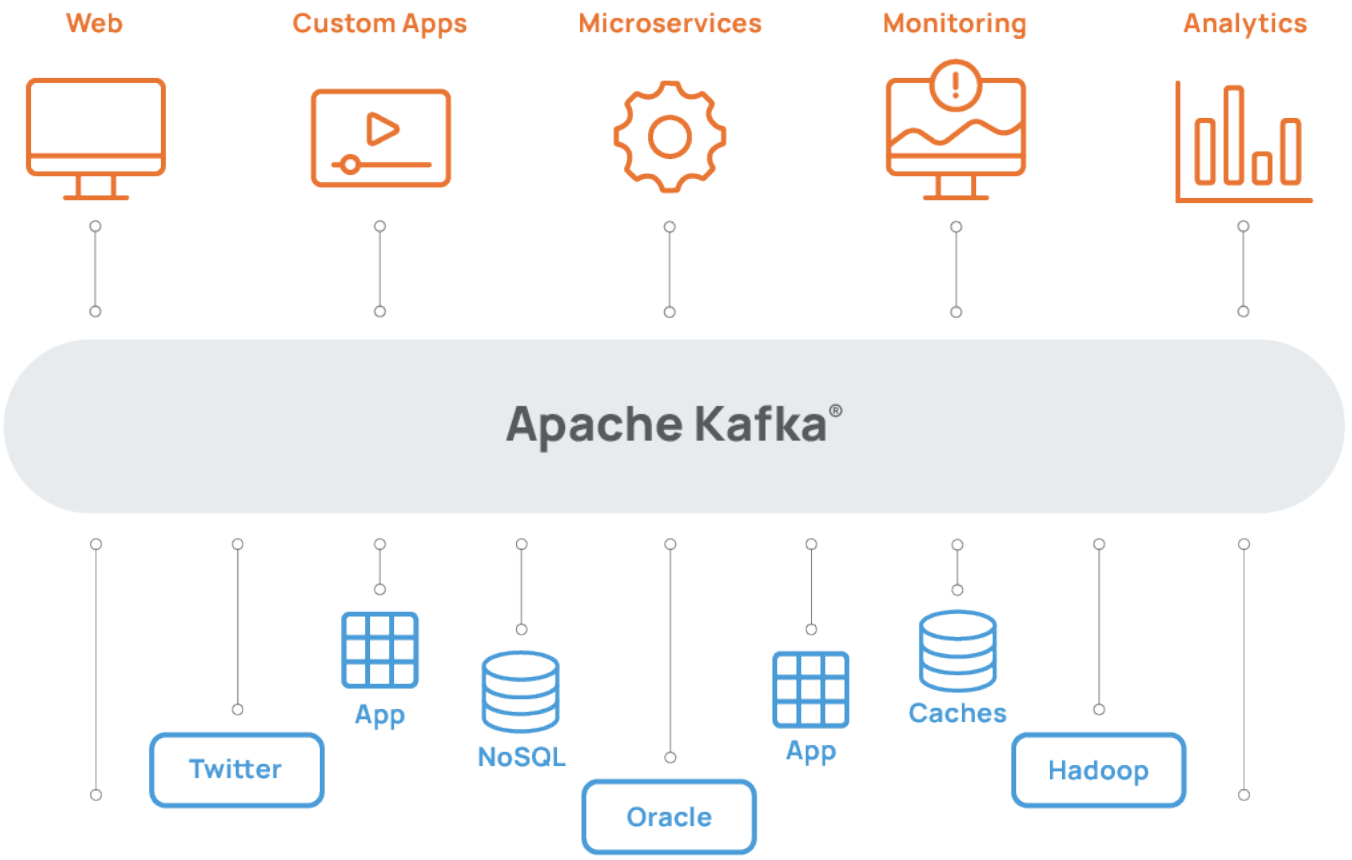
acceldata

How Continuous Data Observability Solves Kafka's 5 Biggest Challenges

Enterprise IT environments rely on a massive ecosystem of event streaming, message queuing, and publish-subscribe (pub/sub) software and services. Open-source tools such as Apache Spark, Pulsar, and Flink's Mozilla RabbitMQ function side-by-side with hungry cloud-native startups such as Estuary, StreamSets and RedPanda, as well as solutions from the big tech providers like AWS Kinesis and Google Pub/Sub.

The unquestioned apex of this real-time streaming food chain is Apache Kafka. There are many reasons for Kafka's popularity, including the savviness of its highest-profile commercial champion, Confluent, but on the technical side, Kafka's success is generally attributed to these five reasons:

- Massive scalability *(read my post on how Big Tech firms like Uber, LinkedIn, Walmart and Tencent process trillions of messages daily with Kafka)*
- Ultra-fast performance
- High reliability
- Huge selection of compatible data and analytics tools
- Extreme flexibility with a large number of options



However, achieving the promise of Kafka has never been a cakewalk. The flipside of Kafka's flexibility and selection of third-party tools is its incredible complexity that trips up new and experienced Kafka admins alike.

Kafka routinely gets compared with another notoriously difficult and labor intensive big data platform, Hadoop. This reputation has scared many organizations into delaying their Kafka rollouts, or has driven them to embrace less intimidating real-time alternatives.

This guide describes the seven most-common technical issues with Kafka that after more than a decade, continue to confound data engineers and developers, and scare users away.

It also explains how a continuous data observability platform like Acceldata can help manage Kafka complexity, and offers specific examples of powerful Acceldata features that empower admins to efficiently manage and monitor their Kafka clusters and optimize their cost-performance.

Challenge No. 1: Kafka is Complex to Setup and Manage

By all accounts, deploying, managing, and optimizing Kafka is extremely difficult and labor-intensive. Its overabundance of low-level manual options — "too many tunable knobs," declares Kafka expert Emil Koutanov — makes Kafka "a beast to learn" and "overwhelming…for newcomers but also seasoned pros."

Why is Kafka so complex in this era of low ops and automation? Because it was originally created by a Silicon Valley giant, LinkedIn, for its own fast-growing internal needs. To futureproof Kafka, LinkedIn's engineers designed it to be the ultimate in scalability, fault tolerance, performance, and tune-ability. In other words, LinkedIn built the equivalent of a million-dollar Italian supercar in 2011 to ensure it would still be top-of-the-line a decade later and serve its own needs (as of 2019, it had more than 100 Kafka clusters with 4,000 Kafka brokers (servers) streaming 7 trillion messages per day).

Over time, maintaining Kafka's millisecond-level transmission speeds and near-zero downtime requires constant monitoring and low-level reconfiguration. This is not a problem for LinkedIn, with its huge army of data engineers and IT administrators and decade-plus of best practices.

For the other 99 percent of companies, this creates an overwhelming operational burden due to what Koutanov calls Kafka's "appalling" native monitoring and management tools. "It's like buying a Ferrari, only to have it delivered with plastic hub caps," he writes.

And while Kafka is awash in features, it cannot handle a real-time data pipeline end-to-end on its own. It doesn't transform data or stream batch data well. It's an inefficient way to store data and poor at serving queries. It is merely one layer in the modern real-time data stack, meaning it must connect to other tools, of which there are many options.

"Kafka is inherently hard to implement and maintain, to the point where many organizations fail to meet their goals," declared a recent blog on Towards Data Science. "On top of that, it brings in additional complexities when integrating with client systems. These characteristic challenges create a high barrier to entry and ongoing maintenance headaches."

How Acceldata Can Help: Our platform provides a unified solution for continuous visibility into Kafka and the rest of your dynamic data pipeline. Acceldata provides best practice configuration recommendations and alerts you when issues do arise from misconfigured servers. Acceldata also enables you to predict and prevent looming issues. And it empowers admins to easily and safely cost-optimize Kafka. Combined, this enables organizations to efficiently manage even disparate Kafka clusters without a legion of data engineers.

Challenge No. 2: Scaling Kafka Kills Your Performance

Kafka is rightly famous for its scale and speed. According to Confluent benchmarks, a single Kafka cluster made up of three broker servers can stream up to 200,000 messages per second, or nearly 67,000 messages per second, per broker. Chinese social media company Tencent streams 10 trillion messages a day with 1,000 brokers, for a whopping 348,000 messages per second, per broker.

While adding a single new Kafka broker to a cluster is easy using default settings, there's a good chance your new broker will be so slow that it will drag down your cluster's overall performance.

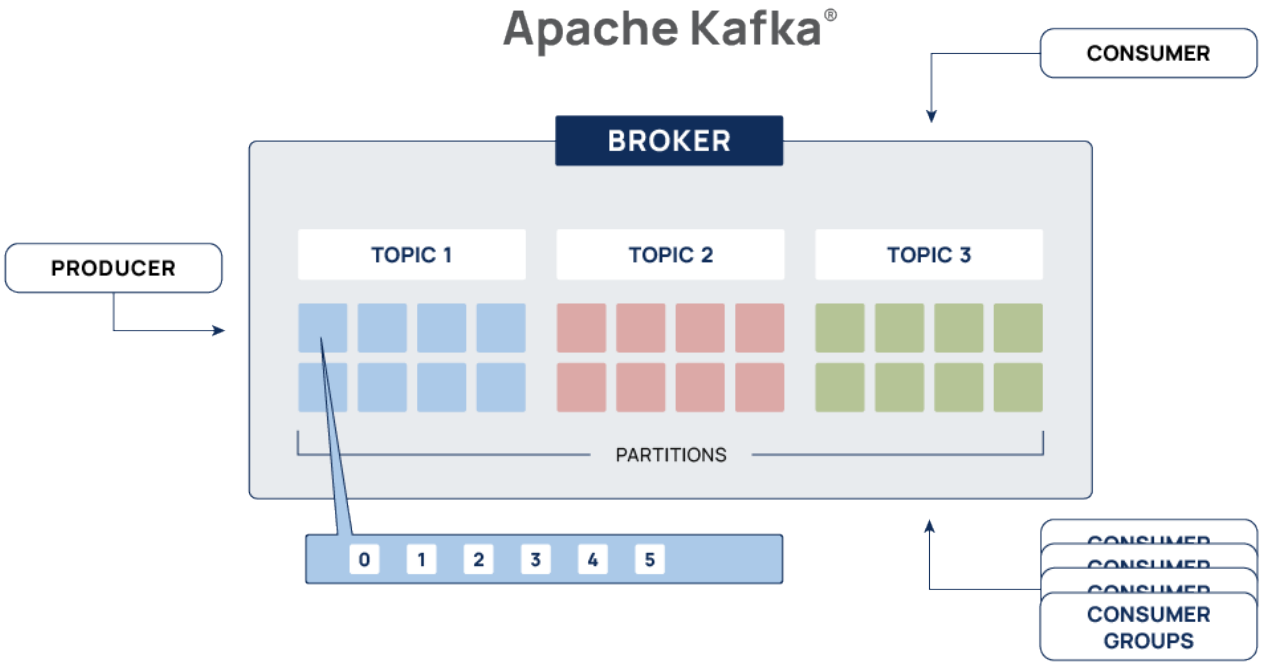
The reason? By default, a broker is set to have only a single partition per topic. Partitions slurp up RAM, so minimizing partitions preserves your server's memory. However, this also constricts your streaming ceiling. To maximize throughput, the number of partitions should be balanced with the number of consumers accessing it, which can be tens or hundreds or more per broker.

Other times, adding a broker to a cluster will create a massive temporary slowdown. For fault tolerance, a new broker by default replicates the topic and partition data from existing brokers. This process of hydrating brokers can take a long time and use a lot of RAM and network I/O if there is a lot of data and/or partitions. And if you are relying on Kafka's native monitoring and management tools, you may not realize the seriousness of the bottleneck, nor have the ability to easily resolve it. And while slowdowns created by new brokers aren't permanent, they can have permanent implications, such as lost data.

Still another example is Kafka's challenges with transforming messages and events on the fly. Any data transformation requires "building a complex pipeline of interactions between producers and consumers" that "significantly reduces" Kafka throughput. Most users turn to third-party ETL/ELT tools. But that forces Kafka admins to test, integrate, monitor and manage yet another tool.

How Acceldata Can Help: Acceldata helps data engineers monitor and fix performance bottlenecks caused by new brokers and other factors that can eventually lead to data loss. Our powerful compute observability solution enables you to maintain a global view of brokers that are failing or slowing down and topics that are becoming skewed.

As for data transformation, Acceldata has native integrations with the most widely-used data solutions so that users will still enjoy a single, unified view into their entire data pipeline. This gives you the confidence to connect Kafka to ETL and other third-party tools and services without fear of creating a data blind spot.



Challenge No. 3: Data Loss from Consumer Lag

When you boil it down, Kafka has the ability to be quite simple from an operational level, if the right data observability and management is applied. Data is streamed from a source (producer) into a server (broker) where it is stored temporarily as a topic and then sent onward via partitions to an application or database (consumer).

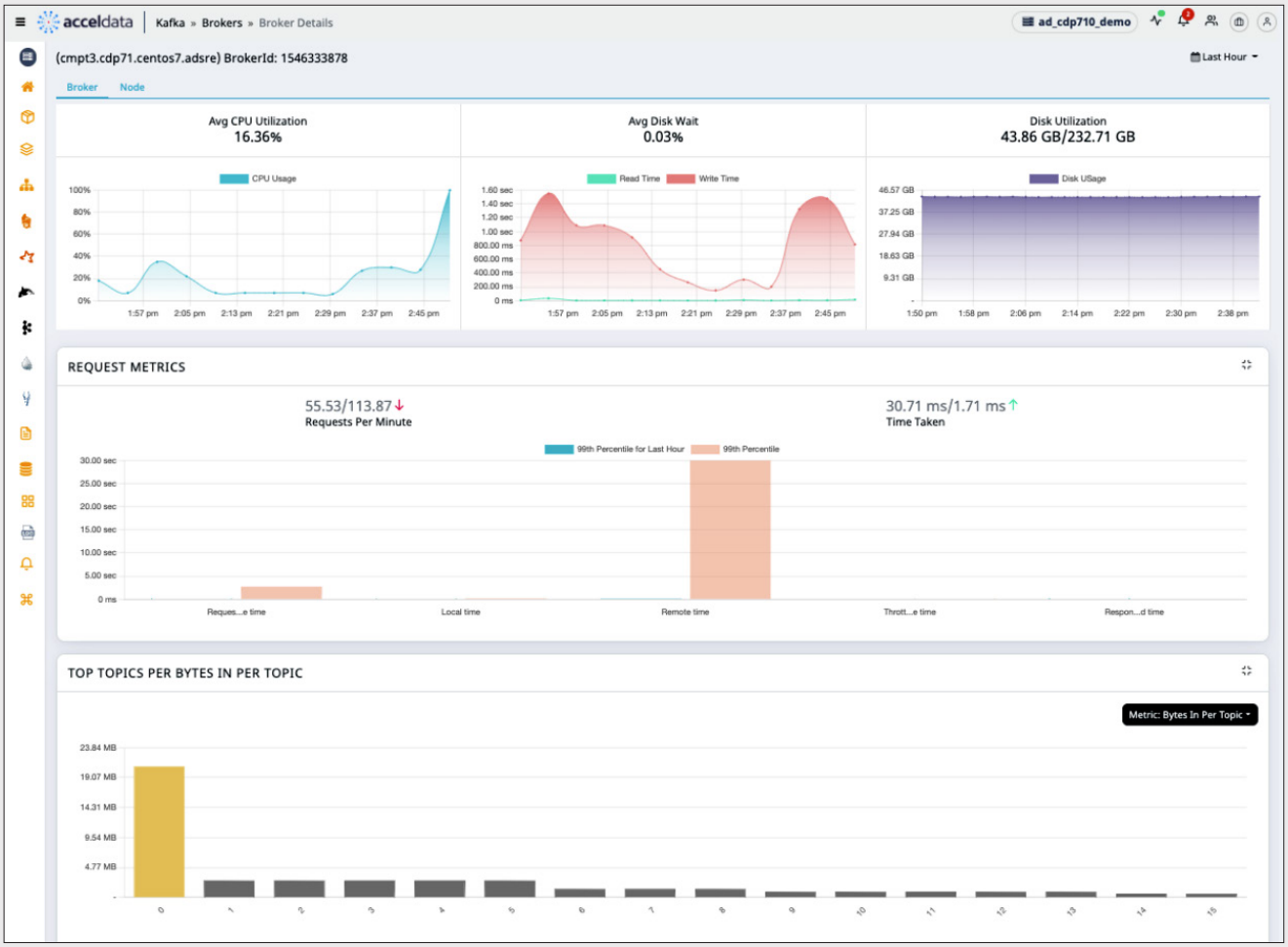
The key word above is temporarily. A Kafka queue is meant to be in constant motion. And brokers aren't true databases. While data is persisted to disk, the default retention period is just seven days. Admins with big storage budgets can keep data for months or years in Kafka, but that is very rare.

So when your Kafka pipeline gets unexpectedly clogged, unsent data builds up and gets purged before it can be sent. New brokers can create this consumer lag. An even more common cause is a poorly-optimized Kafka cluster.

Optimizing your Kafka cluster is a delicate balancing act. It's easy to tune your Kafka cluster incorrectly, or for your cluster to go out of tune over time. For instance, more Partitions typically boosts topic → consumer throughput. But that also increases the chances of a broker server failure. Increasing the maximum CPU usage for a broker can save on the cost of an additional Kafka server. However, it also increases end-to-end data latency and the number of unbalanced topics. Both increase the risk of archived data expiring before it is sent.

Less experienced Kafka administrators relying on the seven day default might be prone to such problems. But even old Kafka hands can be vulnerable if they lack good visibility into Kafka and/or the ability to quickly implement fixes such as scaling up or down different parts of the Kafka pipeline.

How Acceldata Can Help: Our unified dashboard allows you to efficiently oversee every broker, topic, controller, and partition in your Kafka environment. Moreover, our alerts will let you know where the largest consumer lags are emerging. .



For instance, if Kafka is set for the default seven day data archive and a consumer is already lagging by four days, then Acceldata can proactively send an alert to the Kafka administrator to take action before any data is lost.

Challenge No. 4: Kafka's Defaults Undermine Data Reliability

Kafka has a strong reputation for data reliability for four reasons. First, before Kafka, most message queuing systems did not persist data at all, just streaming it through. Many still do not. Second, Kafka's fault-tolerant design means that brokers are constantly backing up data from each other so no messages are lost in case of a crashed server.

Third, Kafka guarantees that messages will be streamed to consumers in the same order they were ingested from producers. And finally, Kafka offers three increasingly-strong options for message delivery:

1. **At-Most-Once delivery,** in which messages are sent only once, meaning errors or disruptions can result in non-delivered messages;
2. **At-Least-Once delivery,** in which Kafka will resend the message until it receives a notification from the consumer guaranteeing a successful delivery (with the possibility of a duplicate delivery);
3. **Exactly Once delivery,** in which Kafka will resend the message until it gets confirmation of receipt by the consumer, while ALSO taking steps to ensure a particular message is not delivered and written more than once.

All sounds great, right? However, each positive also has a pitfall. We already discussed the potential issues with Kafka data retention and broker auto-replication. For Kafka's message order guarantee, the caveat is that this guarantee only applies per partition. Unfortunately, most Kafka brokers each host tens or hundreds or more partitions (LinkedIn has an average of 1,750 per broker). Also, events in Kafka are not tagged with a unique producer ID by default. That means messages and events can still be delivered out of order to consumers, which may lack the tracking information to know.

Finally, on the message delivery guarantee, the safest option, Exactly-Once delivery, requires the most compute and network bandwidth. Since CPU and bandwidth translates to costs, many Kafka admins choose At-Least-Once delivery or the weakest and default option, At-Most-Once delivery. That means if there is a transmission or software glitch, a message can either be lost or received more than once.

While admins can turn on unique Producer ID tags and Exactly-Once Delivery, this can be easily overlooked among the many other settings they must manually configure.

How Acceldata Can Help: Acceldata's data pipeline observability was recently augmented by our custom Kafka utility called Kapxy, which tracks Producer-Topic-Consumer lineage. This delivers a high-level end-to-end view of Kafka data flows, and allows admins to drill into specific messages to ensure they are delivered and not lost or duplicated.

Challenge No. 5: Managing Costs in Kafka and Confluent Cloud

The absence of native Producer-Topic-Consumer lineage doesn't just undermine Kafka data reliability and increase Mean Time To Resolution (MTTR). Without this end-to-end tracking metadata, customers also have a much harder time apportioning usage by application or department. This prevents accurate chargebacks, which discourages internal users from reusing existing Kafka pipelines. Instead, they ask engineering to build more one-off streaming pipelines, creating more work and driving up operational costs.

This operational overhead, combined with Kafka's one-size-fits-none default settings that have an annoying tendency to backfire and create lost or unreliable data, has driven many organizations to Kafka in the cloud, especially the fully-managed serverless Kafka offering called Confluent Cloud.

Confluent Cloud slashes customers' administrative burden to nearly zero by automating almost all Kafka monitoring and management tasks. This allows customers to shrink their data ops teams.

However, users are starting to realize that the security blanket provided by Confluent Cloud comes at a price. There's also the cost of refactoring your Kafka environment when you migrate to Confluent Cloud from a conventional on-premises or hosted Kafka. And your monthly bills can spike dramatically as your data volumes due to Confluent Cloud's pay-as-you-go model and instant, automatic scalability up to 10 GB per second. That's something your users may appreciate, but your CFO will not.

Preventing such a cost overrun is not easy. Although Confluent Cloud provides preliminary consumption data in real-time, actual costs don't appear for 6 to 24 hours. Moreover, Confluent Cloud does not allow customers to set real-time warnings for expenses. Users may not realize the problem until their monthly bill arrives. Which may be why Confluent actually encourages customers serious about real-time monitoring to use a third-party provider.

How Acceldata Can Help: Acceldata's cost-performance dashboards have been augmented by our custom Kafka utility, Kapxy. This gives Kafka users granular visibility into data pipelines that can support ROI and chargeback initiatives, value engineering efforts, and efficient cultures of data self-service and reuse.

As for Confluent, we augment Confluent's native cost and performance metrics with our own monitoring, and then combine the two for additional insights. This allows us to set up real-time alerts around rising costs, and proactive recommendations on realigning your infrastructure to prevent cost overruns.

acceldata

Solving Kafka - Case Studies

PubMatic

PubMatic is one of the largest AdTech companies in the United States. Its data stack included 50 Kafka clusters, each between 10-15 nodes in size. Due to the scale of its overall infrastructure, PubMatic was suffering from frequent outages, performance bottlenecks, and slow MTTR. After deploying Acceldata, PubMatic improved its Kafka data reliability and performance, and consolidated its Kafka (and Hadoop servers) to save millions in software licenses.

ORACLE

For enterprise software giant Oracle, Acceldata identified performance issues in Kafka and made optimization recommendations that improved Kafka's runtime and resource costs by 40 percent, exceeding all SLAs.

PhonePe

PhonePe is India's largest mobile payments company with more than 350 million customers. The company was suffering from constant scaling and performance issues in Kafka and Hadoop. Using Acceldata, the company can now monitor its entire data stack in real-time via automated alerts and easy-to-read dashboards. And when problems emerge in Kafka or Hadoop, PhonePe's engineers can identify root causes faster than before. PhonePe now maintains 99.97 percent availability while scaling its data infrastructure by 2,000 percent.

true

Asian telecom True Corporation was suffering similar performance and scalability issues that left half of the data streamed from Kafka into Hadoop unprocessed. After deploying Acceldata, True Digital boosted its Kafka performance, including raising its Kafka streaming volume to 69,000 messages a second.

Schedule a demo

to see how Acceldata can help you easily and efficiently manage Kafka and the rest of your modern data stack.