Protecting Data Integrity in Confluent Cloud: Over 8 Trillion Messages Audited Per Day

Technology > Confluent JUL 30, 2021 READ TIME: 9 MIN

It's about maintaining the right data even when no one is watching.

Last year, Confluent announced support for Infinite Storage, which fundamentally changes data retention in Apache Kafka® by allowing you to efficiently retain data indefinitely. We're seeing a shift in how users think about the role that Kafka plays in their tech stack, mainly using it as a system of record. This gives rise to new challenges and responsibilities to make sure data is durable and safe. At Confluent, we've been hard at work solving difficult and innovative data durability challenges in order to:

- Protect the integrity of our users' data in production
- Proactively detect, mitigate, and/or restore our users' data in production
- Proactively detect potential data integrity issues in new features before they reach our users

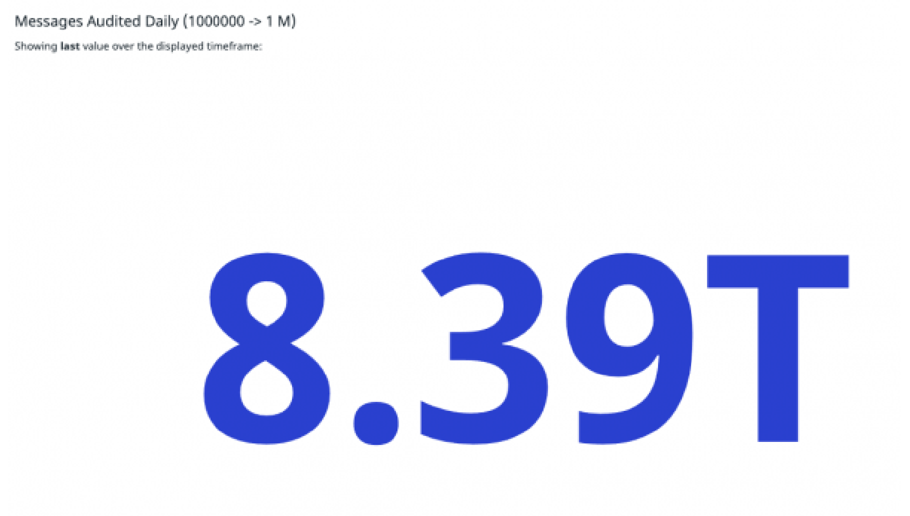
While mitigation and restoration are important topics, this blog post focuses on durability auditing, which is our approach to **proactively detecting data integrity issues on well over 8 trillion Kafka messages per day in Confluent Cloud**.

WRITTEN BY

Rohit Alok Marc

Shekhar Thatikunta Selwan

Staff Product Manager


Messages Audited Daily (1000000 -> 1 M)

Showing **last** value over the displayed timeframe:

8.39T

We’ve all seen the series of 9’s advertised by cloud vendors against their durability SLAs. Those 9’s are intended to provide a guarantee that your data is safe, and a lot of effort goes on behind the scenes to measure and monitor those scores.

Check out the other blog posts in the Design Considerations for Cloud-Native Data Systems Series to learn more about Kafka’s cloud-native capabilities were enhanced for Confluent Cloud:

- Introduction: Design Considerations for Cloud-Native Data Systems
- Part 1: Making Apache Kafka Serverless: Lessons From Confluent Cloud
- Part 2: Speed, Scale, Storage: Our Journey from Apache Kafka to Performance in Confluent Cloud
- Part 3: From On-Prem to Cloud-Native: Multi-Tenancy in Confluent Cloud

What makes durability auditing and monitoring a challenge?

Auditing your data is not only resource heavy but also time-consuming. Hence, frequent audits are tough. This becomes even more of a challenge when you take into account that Kafka is an event streaming platform: Data is always in motion.

Scrubbing, a technique for checking and cleansing data for errors or inconsistencies, is an I/O-heavy operation that can also be time-consuming. This technique needs to continuously run, especially in Kafka, because:

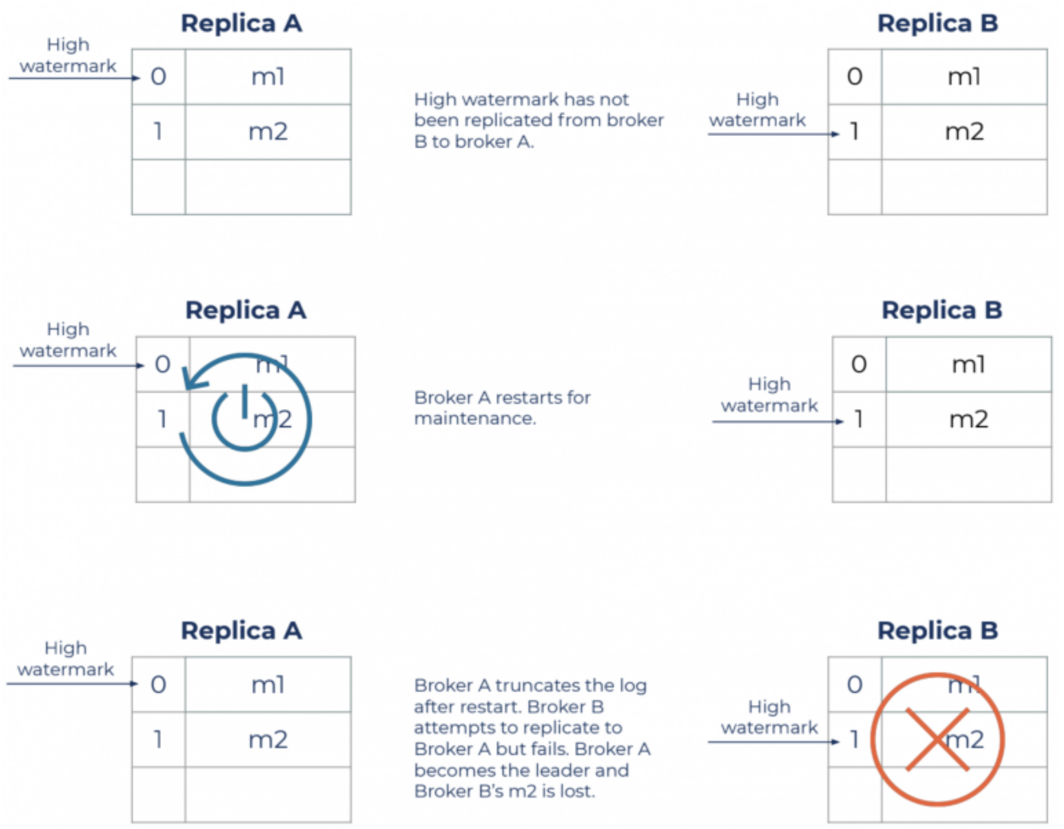
- Even though Kafka records are immutable, what's stored in a Kafka cluster's topic changes constantly as new records are written and, possibly, older records are aged out according to user-defined data retention settings
- Kafka's metadata, such as the epoch, offsets, and high watermark, can change at any time
- Infrastructure or hardware issues can cause data loss or corruption

Challenges from the past – solved today

In the past, there were situations where we were unable to quickly detect durability issues in our own environments. We have taken this to heart when designing our new durability audit service for Confluent Cloud. Keeping our users' data safe is a key pillar for the Kafka Data Platform team at Confluent—we went through this so you don't have to.

In our experience, managing Kafka clusters and trillions of processed messages over the years, we've identified several scenarios where there could be lapses in durability. Some of the notable ones include:

- **Data loss due to replica divergence:** Various complex scenarios, combined with other failures, could cause a divergence in data replicas within a Kafka cluster, leading to data loss.


Replica A
High watermark
0 m1
1 m2
High watermark has not been replicated from broker B to broker A.
Replica B
High watermark
0 m1
1 m2
Replica A
High watermark
0 m1
1 m2
Broker A restarts for maintenance.
Replica B
High watermark
0 m1
1 m2
Replica A
High watermark
0 m1
1 m2
Broker A truncates the log after restart. Broker B attempts to replicate to Broker A but fails. Broker A becomes the leader and Broker B's m2 is lost.
Replica B
High watermark
0 m1
1 m2

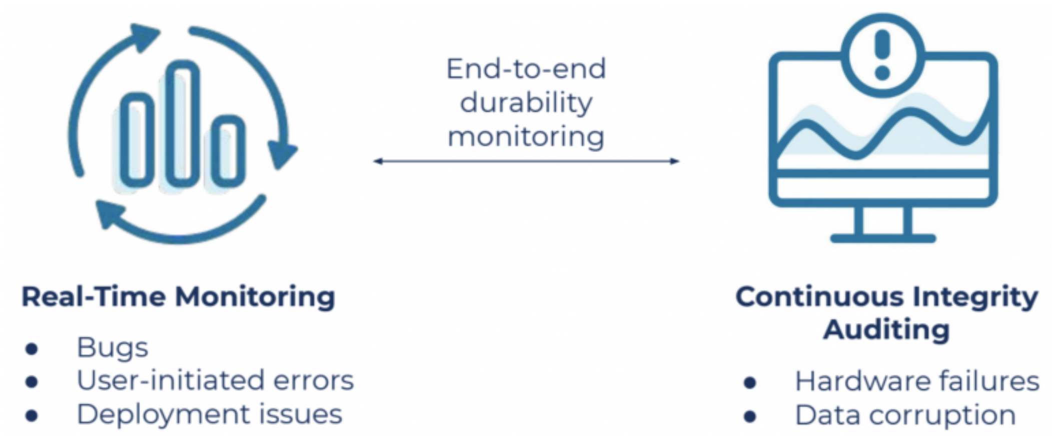
- **Tail data loss:** Storage corruption at the Kafka broker, which is the current designated “leader” of a Kafka topic partition, can cause it to trim the partition (spanning between start offset and high watermark). This forces its followers (other Kafka brokers) to trim theirs based on the leader’s state. This results in data loss even though Kafka’s internal data replication is working correctly between the leader and its followers.
- **Data loss due to metadata divergence:** In our internal test environment, we identified a scenario of data loss caused by a divergence in Infinite Storage metadata triggered by failure to persist an update to broker-local storage.
- **Data loss due to a configuration update bug:** We found a bug when applying Kafka’s dynamic configuration settings, which caused changes in the retention time for some topics.
- **Data loss due to a bug when updating log start offset:** We found an unexpected race condition when updating the log start offset, which causes Kafka to prematurely delete records.

Based on our experience and continuous, extensive testing of Confluent Cloud, there are two main lessons learned:

1. The main causes for data loss/inconsistency are due to software bugs, configuration mistakes, or operator-related errors. Many assume hardware or infrastructure-related issues, but this is not the case because cloud provider infrastructure has high standards for durability, and because Kafka is a distributed, highly available, and replicated system.
2. The key to recovering from, or more importantly, preventing such scenarios is proactively detecting data anomalies. Once they are detected, we implement various methods for data restoration.

Durability

Our approach varies based on the nature of the durability concerns.



Instead of chasing the data, chase the sensitive operations (real-time monitoring).

Any operation that can modify the partition state (metadata or data) is considered a sensitive operation. These operations are a direct result of user actions like configuration change or user-driven deletion of messages, or they are internal to Kafka such as retention management or leadership change. Scrub-based auditing is useful, but in this case, it's simply not enough. It's critical to chase sensitive operations in order to detect issues and alert in real time to quickly restore the state back to normal.

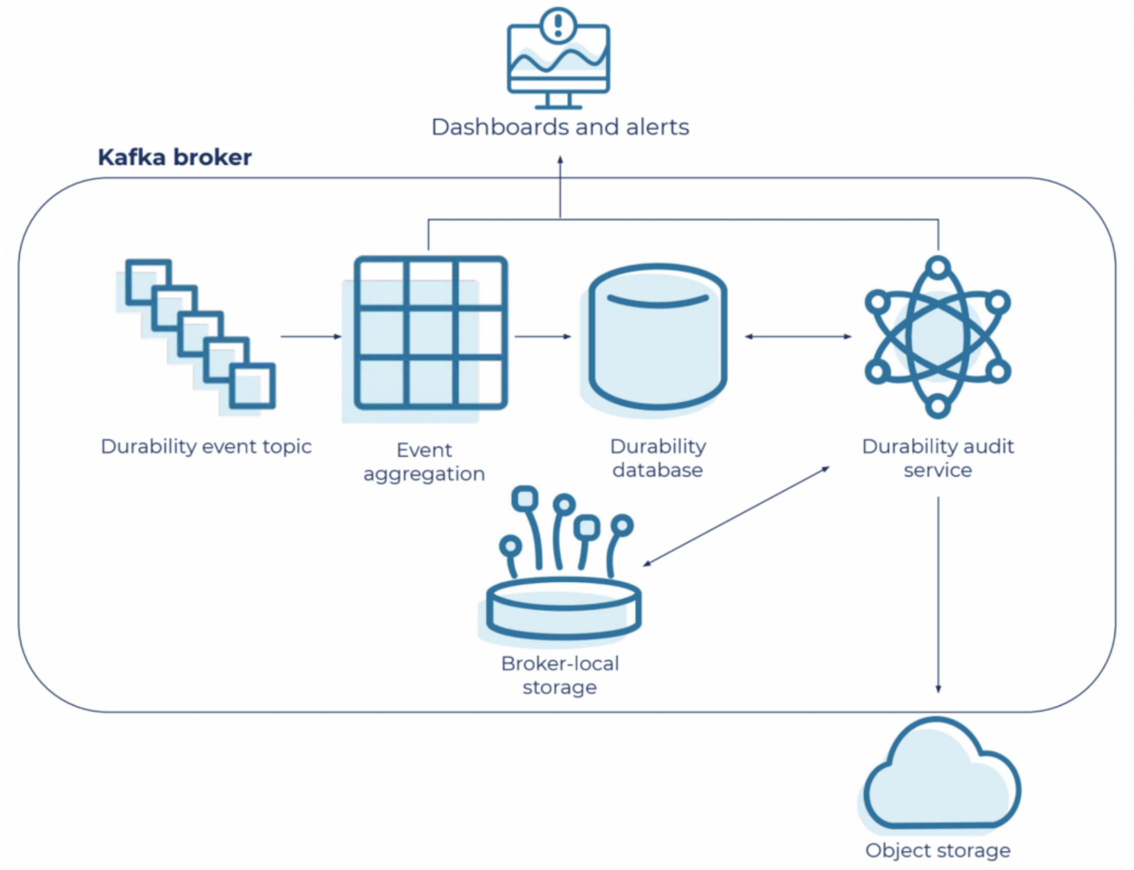
Weighted auditing (extensive integrity audits)

When auditing against infrastructure, there is a big difference in scrubbing metadata (which represents individual segments of the data) versus actual user data as the cost of integrity issues are greater for metadata. For user data, there are additional considerations such as the storage medium used.

Confluent Cloud uses a variety of storage mediums, including storage volumes and object stores, to manage the data that our customers store in their Kafka environments. There is a difference in auditing data on local storage (e.g., Amazon EBS) versus data placed on object stores (e.g., Amazon S3) because each storage medium comes with its own durability guarantees. We use this information to help decide the aggressiveness of our data auditing based on those guarantees.

For example, we consider data (or metadata) on broker-local storage to be more sensitive than the data that we store in object storage due to cloud providers' given durability SLAs. We also consider metadata to be extremely important, so we're even more aggressive with its audit.

The validation of changes in sensitive operations and weighted auditing helps us to perform extensive durability auditing and monitoring with the added benefit of real-time detection and alerting.



Internals of durability auditing and real-time anomaly detection in Confluent Cloud

Source of truth for durability state

It's critical that durability auditing maintains a source of truth, something that can be used to validate the state when sensitive operations occur. A message in Kafka is uniquely identified based on its epoch and offset, which is captured from the partition leader. The metadata is aggregated and materialized in a database, which our durability audit uses as a source of the truth for a given cluster, using the following process:

- We capture the partition leader's sensitive operations, such as epoch changes and high watermark changes
- We guarantee ordered delivery of events

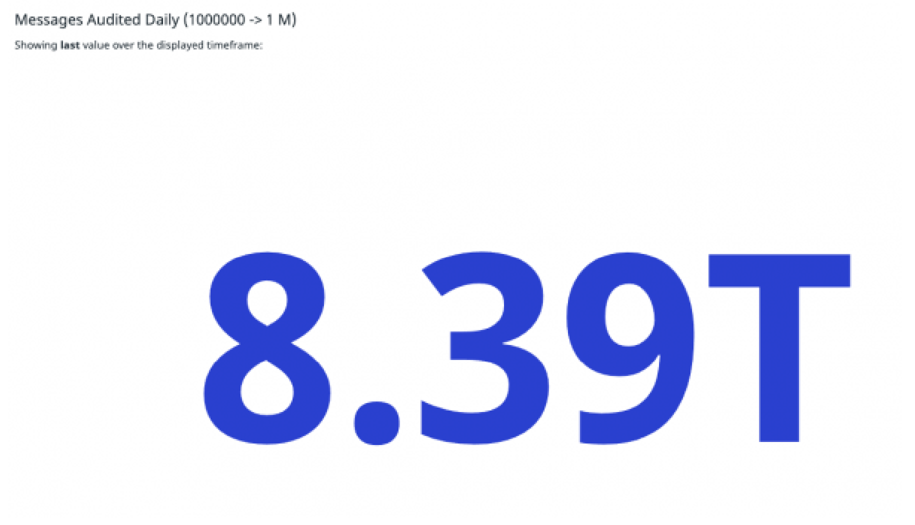
- We then compare these ordered events with our source of truth for real-time anomaly detection
- We also use these events to update our source of truth
- Events like “highwater mark change events” can be generated up to hundreds of thousands of times per second, and to control event logging sprawl, we batch these frequent events into a single aggregated event

Sample of internal state that’s maintained in the durability database.

Durability score

One of the key objectives of a durability audit is to generate scores that can help us understand how we are doing with data durability and how we can continue to improve. The unit of measurement for these scores is the number of messages. The percentage of durable messages are used to calculate this durability score. This means that for a score of 99.9999999, 1 out of 10,000,000,000 messages did not meet our durability checks.

The entire durability audit time frame is divided into spans. A span is a period of time used to audit an entire cluster. During a span, aggressive and sensitive regions may be audited multiple times, while continuous monitoring can alert us to any issues in real time. Once a span is over, the durability scores are updated.


Messages Audited Daily (1000000 -> 1 M)
Showing **last** value over the displayed timeframe:
8.39T

Examples of durability lapses and how we detect them

Tail data loss scenario

An important piece of metadata to Kafka data is the logStartOffset, which is the first offset in a topic-partition. In a real-life scenario, this was accidentally updated to a value greater than what it was supposed to be. This error will cause Kafka to eventually delete messages prior to the logStartOffset.

In this case, our durability monitoring process triggers an event generated from the logStartOffset change. This event is validated against the durability database to detect whether the user deletion or retention policies justify the logStartOffset advancement. We are alerted to this in real time, allowing us to take action before any harm is done.

Replica divergence due to unclean leader election

Unclean leader elections are situations where replicas that are not in the in-sync replica (ISR) are set to be elected as leader as a last resort, even though doing so may result in data loss. Using the figure above for reference, imagine data replica “A” becomes unavailable and an unclean leader election forces “B” to become the leader, which is not currently in sync with “A.” Replica “B” may start epoch 4 at offset 75, which could be inconsistent since replica “A” might have reported some other message at offset 75.

In this case, our durability audit detects the leader election event for replica “B” at epoch 4, causing a validation with our durability database to occur. Since we maintain the historical epoch chain in our durability database, we can alert on this in real time as it’s happening.

Performance impact

When you purchase a cloud service, you’re effectively purchasing a set of performance guarantees (among other important SLAs and limits). It’s important that the protection of a user’s data does not impact their performance expectations. We do this by:

- Using aggregated events to reduce the overall event throughput and load on the system

- Keeping an extremely small and thoughtful amount of metadata, less than 500 bytes, per partition, in our durability database
- Having strong quality of service controls that allow us to dynamically change how aggressive the auditing and monitoring is to make sure that we don't use more resources than allowed

With all of these optimizations, and over 8 trillion messages per day, we confidently assert that:

- The durability audit process doesn't use more than 5% CPU at peak usage
- The storage and memory footprint is limited to tens of megabytes
- The network usage and cloud API calls are highly controlled

Conclusion

Users put their trust in Confluent Cloud with impactful and mission-critical use cases every day. We work tirelessly to make sure that this trust is well placed—the main mission of the Kafka Data Platform team at Confluent. We hope that this deep and technical look into data durability auditing and monitoring shows how committed we are to that mission and that it gives you some innovative ideas on how to protect data in your own systems.

Ready to start using the most secure cloud service for data in motion? Sign up for a free trial of Confluent Cloud and use the promo code CL60BLOG for an extra $60 of free Confluent Cloud usage.*

Get Started

Other posts in this series

- Introduction: Design Considerations for Cloud-Native Data Systems
- Part 1: Making Apache Kafka Serverless: Lessons From Confluent Cloud
- Part 2: Speed, Scale, Storage: Our Journey from Apache Kafka to Performance in Confluent Cloud
- Part 3: From On-Prem to Cloud-Native: Multi-Tenancy in Confluent Cloud

---

Rohit Shekhar is a software engineer on the Kafka team at Confluent, where he works on building the foundations of the next-generation, cloud-native event streaming platform. As part

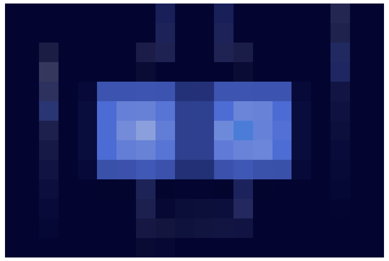
of the initiative, he is leading efforts in providing durability guarantees for Confluent data. Rohit joined Confluent with a successful track record of delivering large-scale data and storage-infrastructure-related features for several successful startups and large-scale companies.

Alok Thatikunta is a software engineer on the Kafka team at Confluent, where he works on building the foundations of the next-generation, cloud-native event streaming platform. He holds a master's degree in computer science from Stony Brook University and an undergraduate degree in computer science from the International Institute of Technology Hyderabad (IIIT Hyderabad). Alok joined Confluent in 2020.

Marc Selwan is the staff product manager for the Kora Storage team at Confluent. Prior to Confluent, Marc held product and customer engineering roles at DataStax, working on storage and indexing engines for Apache Cassandra.

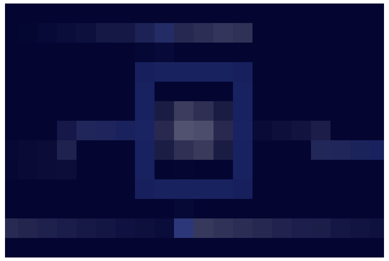
Did you like this blog post? Share it now

Technology < Confluent



Build, Manage, and Monitor Data Streaming Applications, All Within Your Favorite IDE

SEP 17, 2024



Unlock Real-Time Value from DynamoDB Data with Confluent's CDC Source Connector

AUG 27, 2024

We're excited to announce Early Access for Confluent for VS Code. This Visual Studio integration streamlines workflows, accelerates...

OLIVIA GREENE

MATTHEW SEAL

ADI POLAK

62% of Confluent Cloud clusters run on AWS. Meanwhile, hundreds of thousands of customers are using DynamoDB. This blog...

AHMED SAEF ZAMZAM

BRAEDEN QUIRANTE

JAI VIGNESH R

Product

- Confluent Cloud
- Confluent Platform
- Connectors
- Flink
- Stream Governance
- Confluent Hub
- Subscription
- Professional Services
- Training
- Customers

Cloud

- Confluent Cloud
- Support
- Sign Up
- Log In
- Cloud FAQ

Solutions

- Financial Services
- Insurance
- Retail and eCommerce
- Automotive
- Government
- Gaming
- Communication Service Providers
- Technology
- Manufacturing
- Fraud Detection
- Customer 360
- Messaging Modernization
- Streaming Data Pipelines
- Event-driven Microservices
- Mainframe Integration
- SIEM Optimization

Developers

- Confluent Developer
- What is Kafka?
- Resources
- Events
- Webinars
- Meetups
- Current: Data Streaming Event
- Tutorials
- Docs
- Blog

About

- Investor Relations
- Startups
- Company
- Careers
- Partners
- News
- Contact
- Trust and Security