CONFLUENT

Leading Online Delivery Service Increases Messaging Layer Throughput with Event Streaming and Confluent Cloud

Challenge
Increase throughput and reduce latency for interservice communication to ensure seamless transactions at one of the fastest-growing online delivery services.

Solution
Replace Amazon SNS and SQS messaging layer with an event-driven architecture based on Confluent Cloud and Apache Kafka.

Results
- Initial setup time reduced from months to minutes
- Engineering effort reduced
- Operational management minimized
- New capabilities enabled
- Throughput doubled

Online delivery services for groceries and essentials depend on the ability to quickly and reliably complete seamless transactions, starting with a customer selecting the items they want and placing an order, and proceeding through a shopper claiming that order, visiting the selected retailer, purchasing the items, and delivering them to the customer.

At one of the fastest growing delivery services in the U.S., these transactions are powered by an event-driven architecture based on Apache Kafka® and Confluent Cloud. In addition to handling near-real-time event streams for customer orders, this architecture supports a product data pipeline that handles multiple terabytes of data and up to 20 million messages per day. The event streaming infrastructure offers a number of crucial advantages over the legacy messaging layer that it replaced at the company, which was based on Amazon Simple Notification Service (SNS) and Amazon Simple Queue Service (SQS). Confluent Cloud enabled the company to get started quickly, minimize operational overhead, and reduce engineering effort, while providing a cloud vendor-agnostic solution.

"We chose Confluent Cloud so that we could get up and running fast with a hosted Kafka solution," says a senior software engineer for the company. "We started with pay as you go, and had our first service up and running in about an hour. Confluent Cloud enabled us to defer the learning curve on running Kafka in-house. We currently don't have the resources or bandwidth to host and manage clusters ourselves, so instead we have the experts on Kafka doing that for us."

"We started with pay as you go, and had things up and running in about an hour. Confluent Cloud enabled us to defer the learning curve on running Kafka in-house. We currently don't have the resources or bandwidth to host and manage clusters ourselves, so instead we have the experts on Kafka doing that for us."

— Senior Software Engineer

Having replaced SNS and SQS messaging with Kafka and Confluent Cloud, the company is seizing the opportunity to optimize business processes and extend the benefits of event streaming to new initiatives. "We are already launching all new services that we develop on Confluent Cloud," says the engineer. "In addition, one of the big engineering wins we've seen is that we can now convert our in-line product data pipeline into a parallel architecture, which will reduce bottlenecks and get us closer to real-time operations."

Business Results

Initial setup time reduced from months to minutes. "With Confluent Cloud, I had our Kafka cluster set up with simple event publishing in about an hour," says the engineer. "If we had to stand everything up on our own, it would have taken us four to six months, in part because of the learning curve."

Engineering effort reduced. "With the SNS message size limit we faced in the past, we had to write code to split larger payloads and that hurt efficiency," says the engineer. "Similarly, lack of message compression required us to either write more code or pay for more bandwidth. With Confluent Cloud we don't have to worry about any of that because message sizes can be configured and compression enabled with simple configuration changes."

Operational management minimized. "We don't have anyone on staff that's dedicated 100% to managing our Kafka infrastructure, and that's because it's hosted by Confluent," says the engineer. "Confluent Cloud handles our high-volume traffic with reliability and availability—we've had no problems that impacted business since launch."

New capabilities enabled. "Message retention in Kafka and Confluent Cloud enables us to process in parallel all the data in our core pipeline for enriching product information," says the engineer. "Plus, if we find a bug in one of our consumers, we can fix it and then replay earlier messages to resolve any issues introduced. And when we add new consumers to a topic, we have days of retained data to seed them with. None of that was possible before."

Throughput doubled. "Our legacy messaging layer was handling in the neighborhood of 20,000 messages per second," says the engineer. "When we switched to Confluent Cloud, we more than doubled that rate without making any significant changes."

Technical Solution

After using SNS and SQS for several years, the company moved to Kafka and Confluent Cloud to address several pain points with the Amazon services. A key factor was a company-wide push toward cloud-agnostic solutions driven by the CTO. Other drawbacks with SNS and SQS included message size limits and the lack of message retention or message compression, which required the company to expend significant engineering effort in developing in-house workarounds.

Replacing the legacy messaging layer with Kafka removed these limitations instantly. The engineering team now has the ability to configure the message size limit on a per-topic basis and enable compression for any producer to reduce payload size and bandwidth consumption.

The team completed a few test projects with Confluent Cloud before choosing their first production project: replacing an existing service that consumes product data from retailers, enriches it, and then makes it available to other services. "On a typical day, this service processes up to 20 million messages, and we knew that if Kafka worked well for this use case, it should work well for other use cases we had in mind," says the engineer. "It was also a safe place for experimentation, because we could rebuild data stores from the original sources if we made any errors."

Throughout the initial setup and development efforts, the team met regularly with Confluent engineers to talk through best practices and operational details. "We had a helpful monthly sync with Confluent, where we discussed mirroring topics, monitoring clusters, or any other issues we were interested in," says the engineer.

Following their success on the initial project, the team has since added new services to their event-driven architecture and has plans for a larger-scale refactoring of the pipeline. "Message retention with Confluent Cloud has been a nice win, because when we stand up a new service, it can start consuming from an existing topic and the data is already there," says the engineer. "Refactoring the product pipeline by making use of log compaction and moving toward a parallel asynchronous model will be an even bigger win for us."

The team is also working on standard libraries and tools to further speed the adoption of event streaming across the company. This includes an infrastructure-as-code initiative that will enable teams to create topics via Terraform with declarative configuration files.

"With SNS we either had to write custom code for message compression or pay for the extra bandwidth that larger payloads required. With Confluent Cloud, we got that out of the box along with a more flexible and robust solution for interservice communication."

— Senior Software Engineer

Learn More About Confluent Cloud

www.confluent.io/confluent-cloud/