CONFLUENT

Case Study

WWW.CURVE.COM

Curve Opens Up a World of Payment Options with Real-Time, Event-Driven Data Transactions

CURVE

Headquarters
London, United Kingdom

Industry
Fintech

Challenge
While a monolithic architecture was a good starting point for this fintech company, Curve had reached the stage where it was holding them back from truly innovating with all the potential data they had available to better serve customers. The legacy architecture was also creating issues with scalability and server space.

Solution
Curve adopted a microservices-based approach with event-driven data transactions powered by Confluent's data streaming platform.

Results
- Data is more accurate, so customer data is better protected
- Engineering teams can tap into a rich trove of data to enable more and better consumer experiences
- Fully managed Confluent Cloud made scaling, server space, and, most importantly, innovation seamless

Confluent features used
- Dedicated clusters with a private network
- Connectors: PostgresCdcSource, KinesisSource
- Schema Registry
- Lambda and S3 sinks
- BigQuery Connector

The notion of carrying around a wallet full of credit cards seems old-fashioned now. Consumers expect to be able to pay for anything with a flick of a device. Curve takes the concept of easy payment even further, with a mission to simplify the way people spend, send, budget, and save money.

With the Curve card, consumers securely link all their credit and debit cards to one single card or payment app. While the transaction happens on the Curve card, it's immediately passed on to whichever linked card the consumer chooses at the point of sale. Cardholders can even set up rules for certain payment types: one for groceries and another for dining, for example.

With this, a whole suite of fintech capability lands in the hands of consumers. A single mobile dashboard grants a comprehensive view of a user's spending, and they gain full control over managing their accounts—including the ability to go back in time and change payment types.

But efficiency and simplicity for consumers on the front end almost always translates to an extremely complex back end, and Curve is no exception. As a startup, Curve was built on a monolithic architecture, but in order to expand, scale, and compete in a fast-moving market, and continually bring customers the next great feature, the engineering team switched to real-time, event-driven data transactions powered by Confluent.

"Curve has grown to provide many different services, and it made sense to keep those domains separate instead of cramming them into a single monolithic architecture. Making the jump to a microservices architecture allowed us to have multiple teams develop services independently in an agile manner."

— Gustavo Ferreira, Tech Lead Software Engineer, Curve

How to get from a monolithic state to data in motion

In the beginning, like most startups, Curve was mainly concerned with moving fast. "Our goal was to have an MVP—a minimum viable product," explains Gustavo Ferreira, Tech Lead Software Engineer, "and after that, to find a market for it. Things move very fast, and people prioritize moving fast over doing things perfectly."

For this reason, the company's original architecture was a monolithic one. "In my opinion, that's a good choice," Ferreira says. "You don't want to make your systems too complex before you know that the product is going to take off."

At the time, they went with self-managed RabbitMQ as the backbone for asynchronous messaging—a simple message-service bus. As the company grew, this eventually created three categories of problems.

Challenge #1: A setup for failure

The system simply couldn't scale. With only one instance of RabbitMQ, it created a single point of failure, and if that went down, everything went down. The Curve team was spending a significant amount of time on maintenance—and getting paged on weekends because they were running out of disk space. As Ferreira says, "It seemed like every other day, someone would let us know we had too many messages in the queue."

Challenge #2: Hindered innovation

The model made progress tricky for the engineering team. Certain messages were produced to a queue with the intention that only one consumer would be consuming from that queue. In other cases, a straightforward pub/sub mechanism enabled messages to be produced to an exchange and consumed by multiple consumers, without an order guarantee. There was often a lot of re-engineering required of the team to make things work, and as a result, they spent too much of their time trying to find workarounds for architectural limitations, and not enough on straightforward product innovation.

Challenge #3: Obscured data

Batch processing proved to be an issue. The data team needed to build and run one airflow job per database to extract the data and then write it to BigQuery, where the analytics took place. This would take several hours to write.

If the schema changed, it would have gone unnoticed, resulting in missing information. Only when this had been spotted, would a manual backfill be performed.

"Because we had set up the messaging infrastructure ourselves, it wasn't as resilient and battle-tested as using a solution from a third-party provider like Confluent."

— Gustavo Ferreira, Tech Lead Software Engineer, Curve

A world of possibilities with a data mesh

The IT team at Curve considered outsourcing their RabbitMQ efforts to a third-party vendor that could fully manage them. But running RabbitMQ wasn't the only issue or even really the crux of the issue; the problem was what the technology could enable in the first place. What was really needed was a cloud-native, decentralized data streaming architecture.

But while deploying Apache Kafka® is relatively easy, engineering teams often run into growing pains and hidden costs when they try to scale. In addition, the team wanted capabilities that couldn't come from Kafka alone. So they decided to go with Confluent in order to build a true event-driven architecture, gain flexibility with the ability to turn on new clusters quickly, scale easily, and, eventually, enable a data mesh.

With data streaming in place and connectors flowing data from all different areas of the organization, one team can now publish events, and another team can subscribe. This gives project teams the ability to unlock data and create new features quickly. This setup supports various engineering and business at Curve:

1. **Pay tech:** The ability to talk to payment gateways and the logic to process the transactions—both historic and new events
2. **Product:** A world of potential when it comes to building new customer capabilities
3. **Credit:** Best-in-class services for providing customers with innovative credit and lending facilities
4. **Data:** Confluent for ETL pipelines, from data source to the data warehouse all the way through processing

If one team wants to provide insights into consumer spending patterns—how much a user spends on food versus entertainment, for example—that information already exists in the data streams and is easy to tap into. As Ferreira describes it, "These are things that can get unlocked and brought to market relatively quickly if we can offload non-functional requirements to third-party providers as much as possible and instead focus on what we are here to provide—financial products and services."

With Apache Kafka a critical technology, and Confluent the driver, the technology stack at Curve now utilizes Confluent Cloud on AWS, with dedicated clusters on a private network. Various connectors such as PostgresCdcSource and KinesisSource flow the data from the sources to Lambda and S3 sinks. And Schema Registry ensures data quality.

Importantly, the advantages of an immutable durable log, which Confluent provides, allows Curve to do a lot of things the team couldn't do before, such as create entity topics, turn on compaction, and save storage space. With Confluent, there's now an ability to replay messages. If there's a bug in the service that's not discovered for hours, the team can go back in time and reset the affected messages. They can also create a new service today and have it grab messages that already existed in a topic—impossible before. As Ferreira sums it up, now, "You never have two different processors for old data versus new data. There's only one point."

Ultimately, Confluent helps the company scale in a much more durable way. This new model of data streaming also enables the team to get more insights from data. "Data means nothing if you can't analyze it," said Ferreira. "Our company is very data-driven, and almost every decision is based on data."

"These are things that can get unlocked and brought to market relatively quickly if you can focus on the functional side. That's a huge selling point."

— Gustavo Ferreira, Tech Lead Software Engineer, Curve

What's next for Curve?

Curve's database modernization has unfolded over the last few years. "We're still at the very beginning," Ferreira says. "There's much more to explore under the concept of the data mesh."

For one thing, there are still old services on RabbitMQ that would be better suited to a full event-driven architecture, although no new services talk to it. "Eventually, they will need to be extended to talk to Confluent Kafka so we can truly leverage the power of data meshes," Ferreira says.

The next infrastructure project will likely be a data catalog. Curve now has numerous topics, but not everyone knows they exist. Discoverability in the form of a catalog will enable product teams to take full advantage of all the readily available streaming data. And with that will come more new products and features to keep pace with—and even help set—the evolution of consumer banking expectations.

Business results

- **Nimble, scalable access to data**
  Where data used to be bottlenecked and confined, it now flows through connectors, available to all teams so new products and features can be created faster.
- **Reduced development time**
  The ease at which new topics can be added to the BigQuery connector and the automatic schema changes reduces development and support time.
- **Higher consumer security**
  With data available in near real time, data is always accurate, keeping consumer data better protected.
- **More insights from better data**
  Ease of streaming new topics into BigQuery and ensures the data team is empowered to glean stronger insights and build processes to react in real time to what is happening in their environment.

Technical results

- **Much higher availability of data for various teams**
  Data streaming and Confluent connectors give teams across the organization access to all of the data for use in specific and varied developer projects.
- **Easy to update topics**
  If a new topic is added to Confluent, it's as simple as updating the BigQuery connector configuration to include that topic. The table is automatically created and maintained in BigQuery, from where it's joined with other data sources.
- **No more scalability issues**
  With Confluent Cloud, storage for data is scaled on demand, so Ferreira's team no longer has to manage it.
- **The ability to replay data**
  This creates many rich capabilities. For instance, you can fix a bug, and then go back and reprocess the affected data from a few days ago.
- **Maintenance is off the table**
  "We don't even know when the maintenance windows are—and we don't care," Ferreira says.

"If you want more compute, it just shows up out of nowhere. With Confluent, it's taken care of."

— Gustavo Ferreira, Tech Lead Software Engineer, Curve

Learn More About Curve

www.curve.com