7. Exploring the Kafka Ecosystem and Its Future

Contents

[1. Apache Kafka's Success and Challenges 1](#_Toc129946896)

[2. Challenges and Solutions for Data Governance 1](#_Toc129946897)

[3. Challenges and Solutions for Consistency and Productivity 1](#_Toc129946898)

[4. Challenges and Solutions for Fast Data 1](#_Toc129946899)

[5. Apache Kafka's Ecosystem and Summary 1](#_Toc129946900)

# 1. Apache Kafka's Success and Challenges

=>slides: Pg. 1

At this point, we've covered the major components of Apache Kafka. I hope you're feeling equipped with enough knowledge to really start exploring and building big data solutions using Kafka. This module is about taking a step back and surveying the landscape in which Apache Kafka exists. We will discuss the success it has enabled to continue challenges it faces and how it is evolving to meet those challenges head on.

=>slides: Pg. 2

The main use cases for Apache Kafka today have more or less remained the same since it was first created by LinkedIn. It's hard to go anywhere and have discussions about the challenges of data management in modern day enterprises without the mention of Apache Kafka. It is generally regarded as a primary solution for connecting disparate sources of data. With its flexible client APIs, it is possible to write data connectors and syncs for practically any data source. Many of these have been shared and commercialized at this point, and we'll discuss more about them in the coming slides. Apache Kafka is becoming the de facto option for building data supply chains and pipelines that can displace long‑standing, expensive, and fragile ETL environments. Within this context. Apache Kafka fits really well with other "Big Data" solutions like Hadoop and Spark, amongst others, because of its ability to integrate, move, and store data at massive scale. Essentially, reference architectures for data management has started to become established within the industry, and Kafka is a central piece to many of them.

=>slides: Pg. 3

However, sometimes new solutions introduce new problems and reinforce old unsolved problems. Despite the vast utility that Kafka offers today's organizations, there are still a lot of gaps that the industry is being pressured to solve. For example, having the ability to unmask and manage more data actually makes it harder to govern data and manage its rapid evolution. The commoditization of technology and business specialization demands lower overhead and less investment. So regardless of how useful something in technology is, it will always be a challenge the more inconsistent or costly it is to wield. Data is becoming more and more of a strategic differentiator. In the last five years, there has been an arms race for anyone and anything that can manage more and more data. The next 5 years is going to be all about fast data, how to rapidly gain utility from it, particularly in predictive, deep learning contexts. Over the next few slides, I will use these three challenge areas to describe how Kafka is evolving to address these pressures.

# 2. Challenges and Solutions for Data Governance

**=**>slides: Pg. 4

I'll start off with Kafka's challenges with data governance and evolution. Let's consider the common case of a large and growing network of Kafka producers and consumers. As you know by now, each producer is defining its message contract to publish. You'll recall from module 4 that that contract is based on a fairly rigid type dependant serialization system. We didn't talk a lot about this or nearly in as much detail as I would have liked. But in advanced cases, it becomes infeasible to restrict message contracts solely based on the built‑in serializer types. Eventually, as more data diversity is introduced from different systems, custom serializers come into play. Throughout the message lifecycle, there can be hundreds of different contract versions in motion, with each producer publishing massive amounts of data into Kafka. Of course, it takes consumers to derive any sort of value from the data being produced, but they have to be able to do it by reading the data first, which they're able to do through deserializing the message content. This means that with a growing diversity of producers and the data they're publishing, there is an increased complexity all around because consumers have to work with the data being produced and the specifications for each type of method it's consuming. The challenge with Kafka in this common scenario is the lack of some common means of cataloging, registering, and reconciling the disparate message specifications and compatibility mappings between the serializing producers and the deserializing consumers.

**=**>slides: Pg. 5

Confluent is one of the biggest Apache Kafka ecosystem contributors, and they have recognized the challenge we just covered. Fortunately, they have started to take steps to address it by introducing the Kafka Schema Registry. This welcome addition to the Kafka family deserves its own course because of the richness of its functionality. But for now, let me introduce how it addresses the challenges we just covered. One of the more universal data serialization formats out there today is called Apache Avro. It was created to address the challenges with disparate data formats and serialization schemes that make integration and interoperability difficult. It is a self‑describing version format that has broad industry adoption. With Avro, producers can serialize their messages in an Avro‑versioned and self‑describing format and expect them to be deserialized seamlessly by the consumers. As the name suggests, the schema used by both producers and consumers can be registered and version managed centrally within the Kafka cluster environment, allowing for easy, RESTful service‑based discovery and version compatibility reconciliation. Now the great thing is the source is fully available on GitHub and available through the generous Apache version 2 license.

# 3. Challenges and Solutions for Consistency and Productivity

**=**>slides: Pg. 6

Let's consider a typical enterprise data environment. There are many sources and targets for data. Kafka has made quite a reputation for itself in being the conduit between these sources and targets. But the challenge has been a lot of duplication of effort in terms of writing producer and consumer applications that connect the sources and targets together. The crazy thing is, when you think about the work to integrate data stores, they're all more or less the same. I mean, look at relational database management systems, for example. They've been around forever, and there's only so many mainstream database vendors out there, yet, across the industry, it seems that within every company there's the same duplicated effort to write integrating producers and consumers for those very same data stores. Talk about reinventing the wheel. The same could be said about file systems, NoSQL databases, search engines and even Hadoop, amongst others not mentioned. The challenge with Kafka in this scenario has been the lack of consistency in providing a common framework for integrating data sources and targets. It was always left to the individual engineers to create their own solutions, using the generic producer and consumer client APIs. With each integration effort, there is cost not only to develop, but to maintain, and that isn't a very efficient or even productive use of time or effort to do something so common. Furthermore, not every company has the resources to develop and maintain these things, which are really becoming commodities at this point.

**=**>slides: Pg. 7

With the 0.10 release of Apache Kafka, a new framework and marketplace was introduced to address this challenge head‑on. It's called Kafka Connect and the Connector Hub. As with the case of the schema registry, this new innovation deserves its own course to give it justice. The Connect framework is an API for developers. It is intended to make the job of connecting data sources and targets easier and more consistent. The goal is to standardize on a common approach for integrating diverse data sources with standard producer and consumer applications. This is awesome because writing highly performant and reliable consumers, for example, can be really hard and complex, as we covered in the last module. So having a framework to simplify and standardize this is a huge step forward, Now currently, many of the developers using this framework are those that work for the leading technology data providers who have started to include a Kafka connector as part of their product roadmaps. Oracle and HP are some noteworthy examples of this. Currently, there are over 50 platform connectors available that are designed to connect to many different products and services, and that list is growing. Confluent itself has created many of these connectors, and they also provide an online portal they call the Connector Hub. They invite anyone and everyone to develop and contribute a Kafka connector using the API and that online portal for distribution. As adoption grows, this is bound to drive more consistency and greater productivity in Kafka‑based data integration initiatives. Overall, it's going to get cheaper and faster than ever to get Kafka integrated in enterprises.

# 4. Challenges and Solutions for Fast Data

**=**>slides: Pg. 8

Within the last few years, there has been a lot of hype around predictive analytics, machine learning, real‑time, stream‑based, whatever buzzword of your choosing. There are multiple technology platforms that all propose to offer a unique ability to deliver upon this hype for real‑time or stream‑based analytics. Some of these platforms are legitimate, viable solutions such as Apache Storm, Hadoop, Cassandra, and Apache Spark. Again, Kafka is generally found in the middle. But the problem is each one of these technologies introduces a unique and mostly complex model for development and operation. Each have their own API and cluster‑based management approach to distributed systems. Kafka, itself, as we've covered in this course, has its own API and vast cluster‑based model. So if you have all of these technologies under the same roof, so to speak, that's a tremendous amount of technology to manage and maintain all for the same goal of achieving the ability to process and analyze data in real time. Touching on the last challenge, this introduces consistency and productivity challenges and integrating it all together. With Kafka generally being positioned between these technologies for integration, it would need an army of producers and consumers to keep the streaming pipes flowing. The challenge here is pretty obvious. Now I'm not saying all of these different platforms are present in each environment, but many are because they each have their own strengths and advantages that complement the weaknesses of the other. But regardless of whatever of these systems come and go, one thing is becoming more consistent. And that is the place Apache Kafka finds itself within these organizations. **=**>slides: Pg. 9

The 0.10 release of Apache Kafka was a huge one. In addition to Kafka Connect, a new client library for real‑time, stream‑based processing was introduced. This library is called Kafka Streams. The real value proposition of this is that for organizations that have already made an investment in Apache Kafka, they can now have streaming data capabilities without having to install, run, and maintain all of those different platforms. All they need is their existing Kafka environment. Given everything we've learned about Kafka in this course, I am sure you can see how adding this capability to Kafka wasn't that much of a stretch. I mean, consider what Kafka already does with data in motion and how it does it. This is significant because it doesn't require anything more. I mean, theoretically, Apache Kafka could be the only infrastructure solution required. But in reality, many enterprises have good reason to additionally invest in Apache, Hadoop, and Spark. So it may be that Kafka itself isn't the only big data system in place. But at the very least, it can be the only system needed for stream‑based processing. Regardless, the potential to reduce and consolidate into fewer systems is now a very real possibility. Think of what that can do to lower the initial investment and overall total cost. As I said, Kafka Streams is a client library that works with the Kafka cluster. As you've learned in this course, that's exactly like Kafka producers and consumers. They are client libraries too. And just like we did with the producer and consumer client libraries, Kafka streams can be embedded within Java‑based applications, making the barrier to adopt lower than any other platform offering stream‑based processing. Think of it this way. If you already have producers and consumer applications, why not just extend them with the Kafka Streams library to provide stream‑based processing capabilities all within the same place? This is an exciting area that I hope you'll continue to explore.

# 5. Apache Kafka's Ecosystem and Summary

**=**>slides: Pg. 10

Everything that Kafka is today, and what it will be tomorrow, is made possible through the growing and healthy ecosystem of adopters and source code contributors. These are but a few of the big names that not only have based significant parts of their business on Apache Kafka but also make generous contributions back, allowing all companies and organizations, big and small, to benefit. That's the beauty of the open source ecosystem in which Apache Kafka is firmly placed.

**=**>slides: Pg. 11

In this module we covered the undeniable success that Apache Kafka has had since the beginning. It has enabled organizations to solve some of data management's biggest problems. But as I said, in the process, it has introduced new challenges. Many of the challenges facing Kafka and the data management industry in general stem from the rapidly growing and changing landscape. Data volumes, velocity and variety are increasing exponentially, and as a significant player in this landscape, Apache Kafka can't rest on its laurels. Luckily, with the vast and supporting ecosystem it has, Kafka has evolved to meet these challenges and establish reinforced foundations upon which to build further for many years to come. We discussed some of these recent innovations, like the Schema Registry, Kafka Connect and Kafka Streams. I hope you'll agree with me that there is a promising future ahead for Kafka and the many technology professionals and organizations that invest in it. We have come to the end of this course. I hope you learned a lot, at least enough to continue your journey in learning more. It's always hard to decide where to invest your limited amount of time. I personally faced this challenge, as the course author, in determining what details to focus on and what details to sacrifice, because I wanted you to get the most out of this course within a limited amount of time. I hope I succeeded, but it is hopefully just the beginning for you. I encourage you to continue learning about Apache Kafka and trying it out. It's a solid bet to make as a technology professional.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*