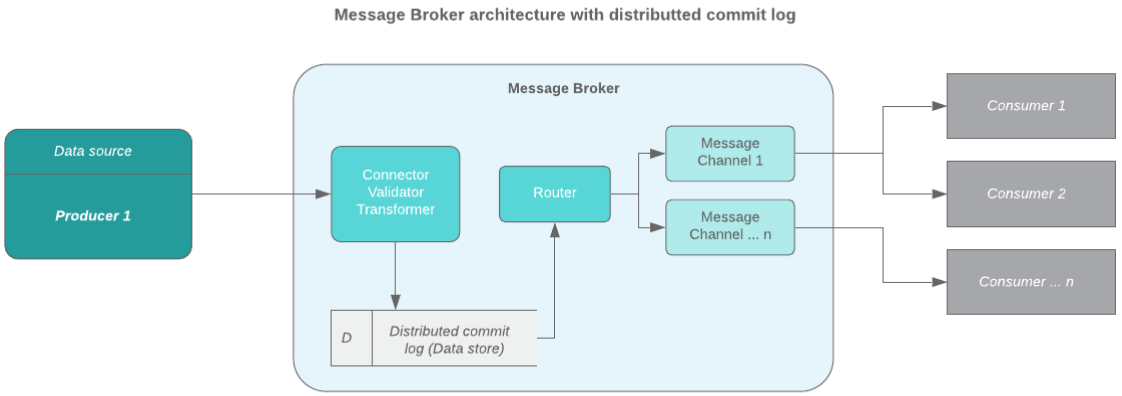
,

AWS Kinesis vs Kafka comparison: Which is right for you?

Throughout the ages, there have always been clashes between great titans, this is also the case in the software industry. We see fierce competition for supremacy by various vendors, each vying for the attention of the consumer space. Two such titans can be found in the field of Message Brokers. They are Apache Kafka and Amazon Kinesis. Recently I was tasked with a project that brought this battle up close and personal. Just when I thought one had a clear advantage and was a shoo-in, the other would come out with unexpected maneuvers that threw the match up in the air. The choice, as I found out, was not an easy one and had a lot of factors to be taken into consideration. So in the battle between Kinesis vs Kafka, the winner could surprise you.

But to understand these titans, we must first dive into the world of **Message Brokers**, we also need to talk about what they are and why they are so important. Message brokers are architectural designs for validating, transforming, and routing messages between applications. It is the middleman between a data streaming source and its intended consumers.

According to Wikipedia - *"The main function of a broker is to take incoming messages from apps and perform some operations on them. Message brokers can decouple end-points, meet specific non-functional requirements, and aid reuse of intermediary functions. For example, a message broker may be used to manage a workload queue or message queue for many receivers. This provides reliable storage, guaranteed message delivery, and transaction management"*

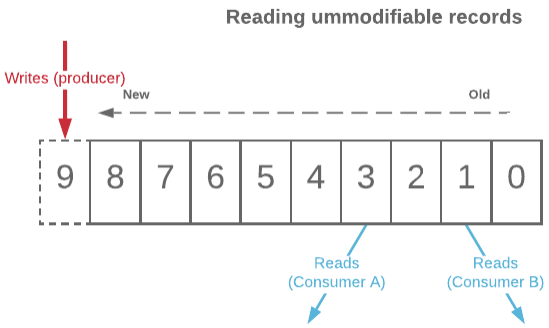


Now you might be wondering why this is so important. Well, a Message Broker is really good at one thing which is processing messages. This means that when you have a lot of messages (thousands, millions, billions of messages) then it could be worth looking into a

Message Broker. It can create a centralized store/processor for these messages so that other applications or users can work with these messages.

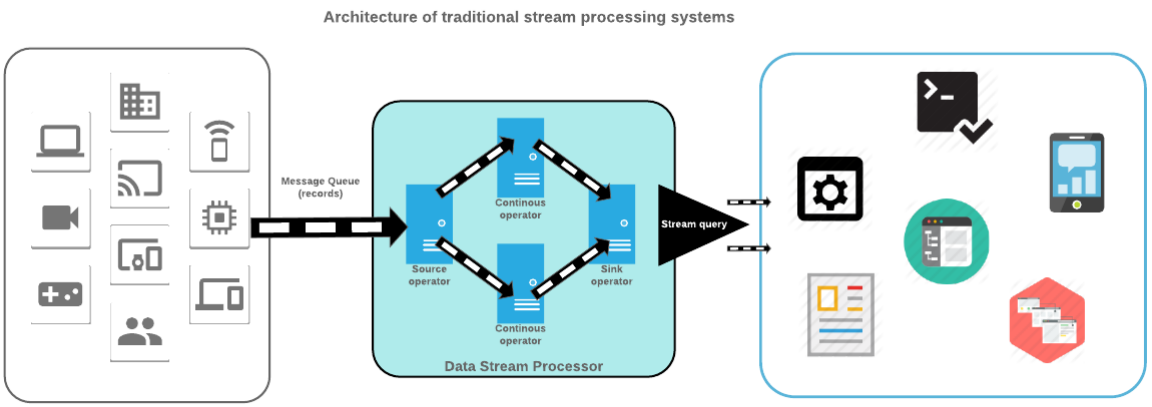
The best use case would be when you have large data streams between applications. You need a middle man to process and direct the data to its intended target. It is also a great solution for integration, especially in Microservices Architecture systems which makes common and standardized data/message bus for all types of apps and services.

As message brokers, Kafka and Kinesis were built as distributed logs. Both do not grant the ability to be modified or changed once an entry has been recorded, while new entries are made only at the end of the log and read sequentially. This gives developers the ability to trace events in the log when there is an issue. Plus the inability to perform modifications increases consistency and security.



Apache Kafka and Amazon Kinesis

To better understand Kafka vs Kinesis, we would next need to introduce **Streaming Data**. This is data that is generated continuously by thousands of data sources. They are sent in the data records simultaneously, and in small sizes (order of Kilobytes). When we refer to streaming data, we are talking about the large collection of generated content. These are gotten from sources such as the web or mobile applications but also e-commerce purchases, in-game activities or the never-ending information generated on social media. Let's not forget that IoT devices are also a source for such large data streams.



What is Apache Kafka?

It is an open-source, high-performance, fault-tolerant, and scalable platform for building real-time streaming data pipelines and applications. Apache Kafka is a streaming data store. It decouples applications producing streaming data (**producers**), into its data store from applications consuming streaming data (**consumers**) from its data store. The distributed nature of Apache Kafka allows it to scale out and provides high availability in case of node failure. Organizations use Apache Kafka as a data source for applications that analyze and react to streaming data.

What is Amazon Kinesis?

It is an Amazon Web Service (AWS) for processing big data in real-time. The key feature inherent in Kinesis is its ability to process hundreds of terabytes of high-volume data streams per hour. These could be continuously captured from sources such as operational logs, social media feeds, in-game microtransactions or player activities, or even financial transactions. Its advantage over previous technology is its ability to simplify the development process of certain apps. This is done with Kinesis' real-time operational decision-making with streaming data.

AWS MSK solution

But there is, however, a third contender. While it is not a standalone platform like Kafka and Kinesis, it is a streaming data service that manages Apache Kafka's infrastructure and operations. This makes it easy for developers and DevOps managers to run Apache Kafka applications on AWS. All without the need to become experts in operating Apache Kafka clusters or having a dedicated team to manage them. While Kafka is highly customizable, it does take a massive amount of effort to maintain and run. Plus it's not something to invest in without proper infrastructure. But Amazon MSK takes care of this loophole. It does this by operating and maintaining Apache Kafka clusters. Plus provides enterprise-grade security features from the start. It has built-in AWS integrations that accelerate the development of streaming data applications. So in the battle of Kinesis vs Kafka, MSK might be the hidden underdog.

Talking Architecture

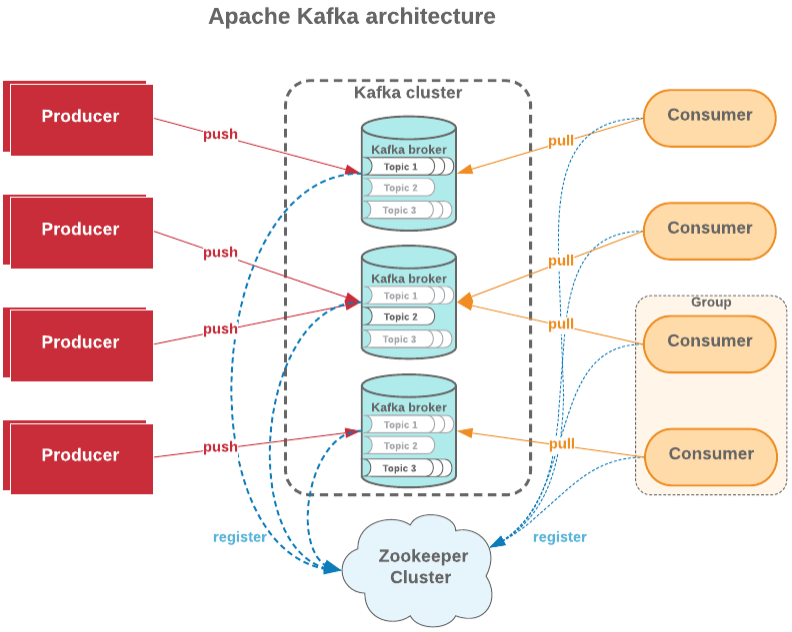
When it comes to core architecture for either Kafka or Kinesis, you will find that although the outcome is similar, they operate very differently.

Apache Kafka is comprised of various components such as Records, Topics, Consumers, Producers, Brokers, Logs, Partitions, and Clusters. Records can have key (optional), value and timestamp. Kafka Records are changeless meaning once written they can not be modified. A Kafka Topic is a stream of records, you can think of a Topic as a feed name. Each topic has a Log which is the topic's storage on disk. Each Topic Log is further broken up into what are called partitions and segments.

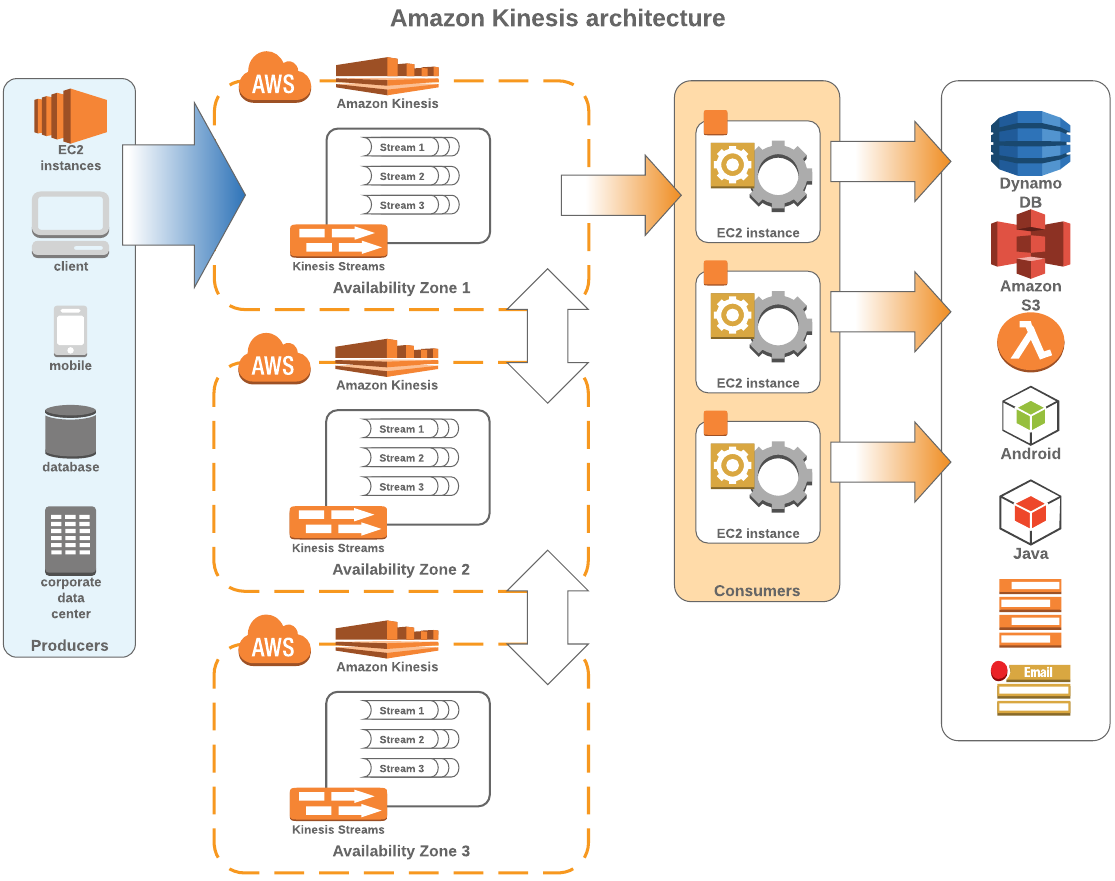
There are four major APIs in Kafka, namely:

- The Producer API: sends streams of data to topics in the Kafka cluster
- The Consumer API: reads streams of data from topics in the Kafka cluster
- The Streams API: transforms streams of data from input topics to output topics
- The Connect API: implements connectors that consistently pulls from some source system or app into Kafka or push from Kafka into others

Next is the Broker which is a Kafka server that runs in a Kafka Cluster. Multiple Kafka Brokers are needed to form a cluster. The Kafka Cluster consists of many Kafka Brokers on many servers. Broker sometimes refers to more of a logical system or as Kafka as a whole.



Kinesis comprises key concepts such as Data Producer, Data Consumer, Data Stream, Shard, Data Record, Partition Key, and a Sequence Number. A shard is the base throughput unit of an Amazon Kinesis data stream. The data producer emits the data records as they are generated and the data consumer retrieving data from all shards in a stream as it is generated. A stream is a logical grouping of shards, while the record is the unit of data stored in an Amazon Kinesis stream. Finally, the partition key is typically a meaningful identifier, such as a user ID or timestamp and the sequence number is a unique identifier for each data record.



Below is a breakdown comparison between Kafka and Kinesis:

	Apache Kafka	**Amazon Kinesis**
Concepts	Kafka Streams	Kinesis Analytics
Stream of records container	Topic	Stream
Data Stored in...	Kafka Partition	Kinesis Shard
Unique ID of a record	Offset number	Sequence number
Ordering under...	Partition level	Shard level
Features		
SDK Support	Kafka SDK supports Java	AWS SDK supports Android, Java, Go, .NET
Configuration & Features	More control on configuration and better performance	Number of days/shards can only be configured
Reliability	The replication factor can be configured	Kinesis writes synchronously to 3 different machines/data-centers
Performance	Kafka wins	Kinesis writes each message synchronously to 3 different machines

Data Retention	Configurable	7 days at max
Log Compaction	Supported	Not supported
Processing Events	More than 1000s of events/sec	Almost 1000s of events/sec
Producer Throughput	Kafka Wins	Kinesis is a bit slower than Kafka
Ops		
Setup	Weeks	A couple of hours
Configuration Store	Apache Zookeeper	Amazon DynamoDB
Checkpointing	Offsets stored in a special topic	DynamoDB
Incident Risk/Maintenance	More In Kafka	Amazon takes care
Human Costs	Require human support for installing and managing their clusters, and also accounting for requirements such as high availability, durability, and recovery	Kinesis is just about paying and use

Courtesy http://www.itcheerup.net/2019/01/kafka-vs-kinesis/

Brief Features comparison

When it comes to features, Kafka and Kinesis offer varying implementations and functions. For example, while Apache Kafka has **SDK support** for Java, Amazon Kinesis supports Android, Java, Go, and .NET. So users of .NET would be more inclined towards tilt towards Kinesis than they would Kafka. But the feature comparison doesn't just end there. While dealing with Kinesis, you would start to notice a bit of limitation on some of its features.

When it comes to **configurations**, Kinesis only allows for the number of days/shards to be configured. Plus you can only write synchronously to 3 different machines/data centers.

In terms of **performance**, Kinesis writes each message synchronously to 3 different machines. This, however, slows down the write operation that in turn affects general performance. Kafka, on the other hand, is more flexible in its configurations. It allows you more control over configuration and better performance while letting you set the complexity of replications. Performance-wise, Kafka has a clear advantage over Kinesis.

Let's not forget that Kafka consistently gets better **throughput** than Kinesis. Kafka can reach a throughput of 30k messages per second, whereas the throughput of Kinesis is much lower, but still solidly in the thousands. While Kinesis throughput improved when parallelizing the producers, in the sense that multiple producers scripts were running in parallel on one machine, it will max out at about 20k msg/sec.

One that can attribute Kafa's supremacy here is its very strong **community** that has been dedicated to its improvement over the years. But we are already seeing improvements in Kinesis as time passes. So we can expect the throughput to increase down the line.

Costs analysis

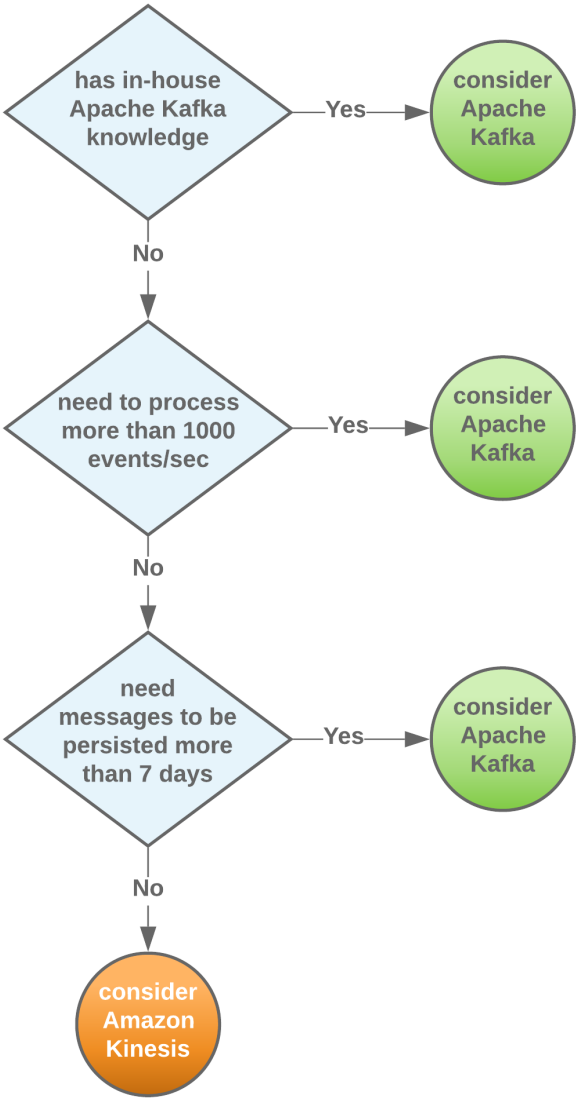
Here is where things get a little more complicated, assuming you are going to run an **in-house Kafka server**. You would think that since Kafka is open source and considered free software, it should not cost anything to implement. This is not the case. What you would be comparing here is the implementation cost of setting up, running, and maintaining a

Kafka installation along with the human resources needed, against the hosted nature of Amazon Kinesis.
To give a clearer picture of what setting up Kafka would entail beyond the downloading of the software, you would need to remember that it is highly customizable. This also means that it's not ready to go right out of the box. A lot of time and effort will be needed to get your installation running. Then there is the added expense of managing and maintaining the installation. If you already have a dedicated team on staff that can handle this, then you can assign the task to them. But for a non-existing team scenario, you would be looking at hiring skilled staff or outsourcing the installation and management. This is both time-consuming and can be expensive. Let's not forget the device cost of what you will be running Kafka on.
Amazon Kinesis, on the other hand, is a simple stress-free process to set up and start using. Once you have paid for the quantity you need, then you are good to go. No hassle or complicated setup. While the Amazon Kinesis is a simple straightforward installation, you will require human resources for its setup In some cases, you can be up and running in a few minutes. It should also be noted that AWS has provisioned-based pricing, meaning you will be charged even if the cluster isn't in use.

Who is using what
According to enlyft.com, there are about 12,792 companies that use Apache Kafka. It also has a market share of about 15.16% which is 10x more than Amazon Kinesis. They stated that: *"Looking at Apache Kafka customers by industry, we find that Computer Software (30%), Information Technology and Services (11%) and Staffing and Recruiting (7%) are the largest segments."*
Amazon Kinesis has just 1% Market share with 478 know sites using it as stated by datanyze.com

Which is right for you
Before running off to pick either of these two solutions, it would be best to note that both are excellent and are very good at what they do. The question though is which is right for you, Kinesis vs Kafka. To answer this you must first take a look at your use case and available resources.
So here is what we can conclude:

- Apache Kafka - gives full flexibility and all advantages of the latest Kafka versions, but requires more effort in its management
- Amazon Kinesis - simplifies the process of introducing streaming technology for those who don't have the the resources to manage Kafka, however, Kinesis has much more limitations than Apache Kafka
- Amazon MSK - an intermediate solution that allows using Kafka as AWS service hence simplifies the setup process and offloads DevOps management, but still doesn't have full compatibility with the latest Apache Kafka versions



If you have the in-house knowledge to maintain Kafka and Zookeeper, don't need to integrate with AWS Services and you need to process more than 1000s of events/second then Apache Kafka is just right for you. You get the flexibility and scalability inherent in the system plus the ability to customize it to your needs.

On the other hand, if you don't have the in-house knowledge to maintain Kafka (a Dev team) or have to integrate with other AWS services such as Redshift, DynamoDB, Lambda, etc plus process 1000s of events/second at most, then getting Amazon Kinesis would be a better choice.

However, not everyone falls squarely into one of these two categories. So a good middle ground using Amazon MSK might be just right for you. You get the flexibility that Kafka gives while also being able to integrate with AWS services. So it may end in a triple duel - Kinesis vs Kafka vs MSK.