



Rediscover
the meaning of technology

Kafka

2022

Hablemos sobre Kafka

Agenda



Introducción a Apache Kafka

Arquitectura y Componentes

Ejemplos

Complementos

Casos de uso

Introducción a Apache Kafka

1. Sistemas de mensajería
2. ¿Qué es Apache Kafka?
3. ¿Qué problema busca solucionar?
4. Comparativa
5. Pros/Contras

Sistemas de mensajería

Sistemas de mensajería

Es una aplicación intermediaria que se encarga de enviar mensajes desde una aplicación (origen) a una o mas aplicaciones (destino).

En este tipo de sistemas mayormente la comunicación se realiza de forma asíncrona.

Las aplicaciones por lo general requieren una infraestructura de mensajería que permita conectar los componentes y/o servicios, idealmente manteniendo un acoplamiento flexible, para maximizar la escalabilidad.

Patrones de mensajería

- Cola de mensajes



- Publicador-Suscriptor



¿Qué es Apache Kafka?

¿Qué es Apache Kafka?

- Es una plataforma distribuida de transmisión de datos (o sistema de mensajería) que permite publicar, almacenar y procesar flujos de datos en tiempo real.
- Es una plataforma de streaming distribuido de código abierto que se puede utilizar para compilar canalizaciones de datos de streaming en tiempo real.

Diagrama conceptual de Kafka

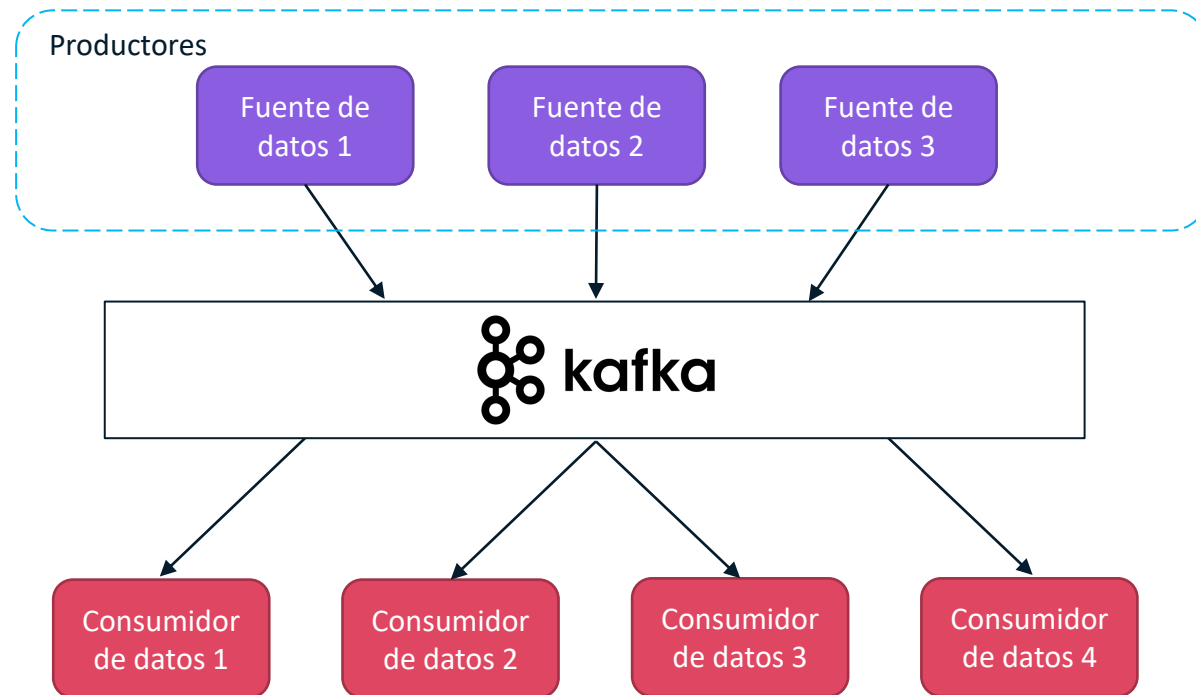
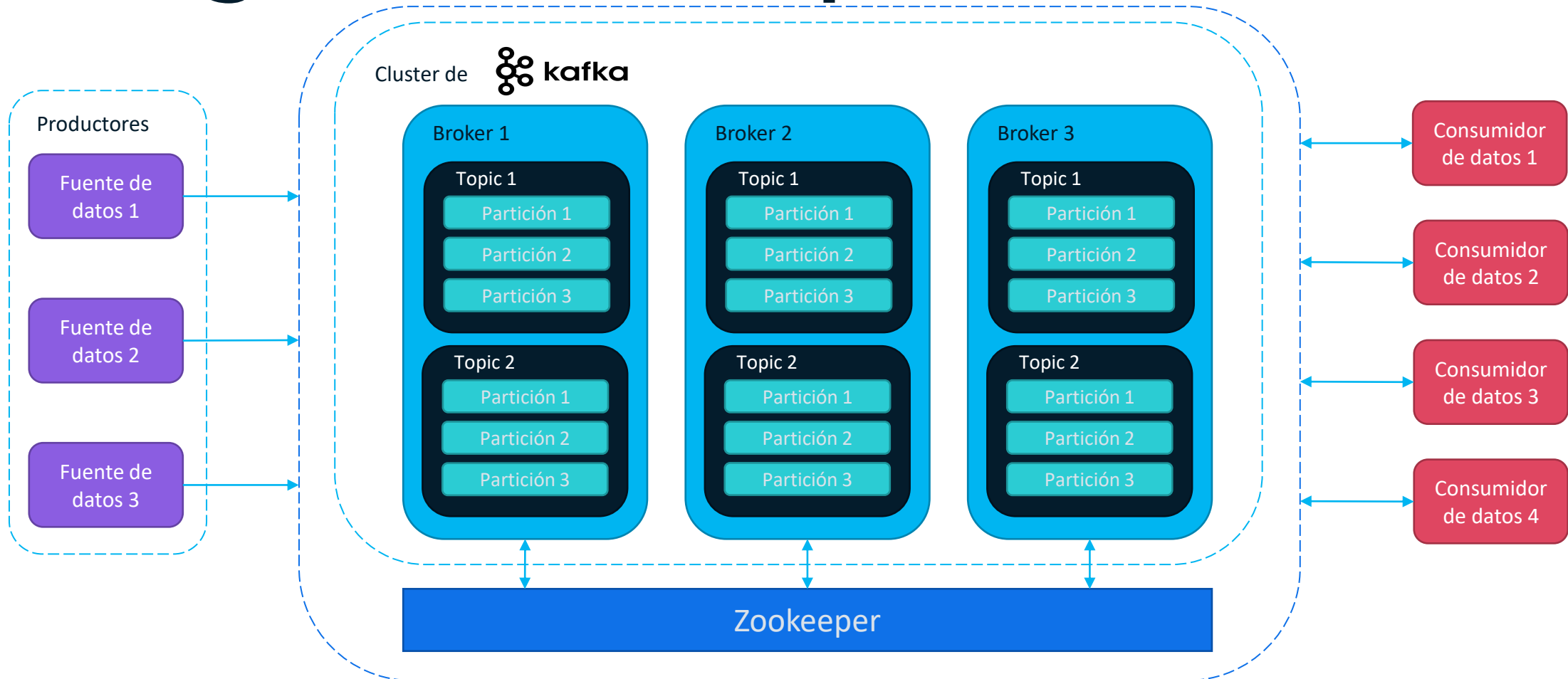


Diagrama conceptual de Kafka



¿Qué problema busca solucionar?

- Desacoplar las fuentes de datos de los consumidores.
- Uso en arquitecturas de streaming de datos en tiempo real.
- Recopilar grandes datos (por ejemplo: sistema de ingesta y gestión de logs, sistema de recomendaciones en tiempo real, tracking actividad de una web, monitoreo de métricas, seguimiento de clicks, etc).

Comparativa



Azure Service Bus



Azure Event Grid



Cloud Pub/Sub



Azure Event Hub



Red Hat
AMQ



amazon
SQS



KubeMQ



Comparativa

	 kafka	 RabbitMQ™	 Azure Event Hub
Tamaño de mensaje o evento	El tamaño máximo por defecto y el sugerido es de 1 MB aunque se puede cambiar	Por defecto 512 MB aunque se puede ampliar hasta 2 GB	El tamaño máximo de mensaje permitido es de 1 MB
Prioridad en los mensajes	Sin función de prioridad incorporada de serie pero existen complementos para hacerlo.	Se pueden usar colas de prioridades.	Se pueden usar colas de prioridades.
Retención de mensajes	Sí es posible utilizando los períodos de retención	No, ya que los mensajes se eliminan de la cola inmediatamente después de la entrega.	Sí es posible utilizando los períodos de retención
Garantías de entrega/procesamiento	Es compatible con los siguientes tipos de garantías: <ul style="list-style-type: none"> • Sin garantía. • Como máximo una vez. • Al menos una vez. • Exactamente una vez. 	Entrega el mensaje sólo una vez	Al menos una entrega de un evento
Licencia/Costo	Código abierto: licencia Apache	Código abierto: licencia pública de Mozilla	Microsoft Azure, pago por uso.
Protocolos soportados	TCP	MQTT, AMQP, STOMP, HTTP	HTTPS, AMQP, Kafka

Pros/Contras

Pros

- Es open source bajo la Apache Software Foundation
- Es escalable, persistente y tolerante a fallos. Lo que lo hace ideal para trabajar con grandes cantidades de datos en sistemas BigData.
- Tiene una baja latencia (del orden de los 5ms), por lo que es apto para escenarios en tiempo real.
- Referencia en la industria, lo usan las empresas como LinkedIn, Airbnb, Netflix, o Google.
- Permite integrar tecnologías de BigData con requisitos elevados de escalabilidad como pueden ser Apache Spark, Hadoop, Apache Flink.
- Garantías de entrega de mensajes exactly-once (exactamente una vez).
- Tiene una comunidad muy grande.
- Soporte empresarial.

Contras

- No es una tecnología que esté diseñada para manejar mensajes muy grandes (+ 1MB). Con mensajes muy grandes se degrada su rendimiento.
- Pocas opciones de monitorización potentes.
- Puede ser compleja su instalación y mantenimiento en múltiples clusters.
- No tienen soporte para mensajes de distintas prioridades.
- Algunos patrones de mensajería no tienen soporte, como son las point-to-point queues, RPC o el de request/reply.
- No soporta el envío de mensajes con comodines en los nombres de los topics (por ejemplo mandar un mensaje a todos los topics o a topic-test-*)
- Dependencia de Zookeeper.
- Si ocurre un fallo en el consumidor después de procesar el registro pero justo antes de enviar el commit al broker, algunos mensajes pueden ser reprocesados.

Arquitectura y Componentes

1. Apache Zookeeper
2. Topics y particiones
3. Mensajes
4. Brokers
5. Replicación
6. Criterios de retención
7. Productores
8. Consumidores
9. Grupo de consumidores
10. Garantías de entrega
11. Transacciones

Apache Zookeeper

Apache Zookeeper

- Es un software que proporciona un servicio de coordinación (mantenimiento de configuración, naming, sincronización distribuida, servicios de agrupación, etc.) de alto rendimiento para aplicaciones distribuidas.
- Almacén distribuido (clave-valor) con las configuraciones de control de la plataforma.



¿Que es Zookeeper para Kafka?

- El encargado de almacenar e intercambiar los metadatos con los: brokers, productores y consumidores.
- Realiza la detección de las cargas de trabajo de los brokers para la asignación de mensajes realizarla por cercanía o por tener una menor carga.
- Responsable de la selección del broker leader donde se almacenará la partición leader.

Topics y particiones

Topics

- Una colección de mensajes conforman un topic.
- Son una agrupación lógica para mantener los mensajes que tienen un mismo significado o propósito bajo un mismo contexto.
- Los topics están divididos en particiones que permiten dividir los datos entre los diferentes brokers para que podamos escalar horizontalmente.
- Los topics se almacenan en disco

Particiones

- Secuencia de mensajes ordenada e inmutable.
- Unidad de replicación o paralelismo de un topic.
- Cada partición se puede hallar en uno o mas brokers diferentes.
- Cada mensaje dentro de una partición tiene un identificador numérico incremental llamado offset.
- Cuando se añade un mensaje en cada una de las particiones se le asigna un offset.

Particiones

- Se establece un broker con partición leader y cero o más brokers con particiones followers "replica".
- El broker leader controla todas las peticiones de lectura y escritura sobre una partición del topic.
- Los brokers followers replican a los leaders y toman el control si el leader establecido "muere".
- Facilita la redundancia y escalabilidad.

Mensajes

Mensajes

- Son cada uno de los elementos que se envían a los topics y son persistidos en el disco de cada broker que los contine.
- Los mensajes siempre se escriben al final de un topic.
- Son inmutables.
- Pueden ser eventos, comandos, documentos.
- Por defecto la retención de los mensajes es de 7 días.
- Cuando ha pasado el periodo de retención son eliminados independientemente de si se han consumidos o no.

Mensajes

- Equivalen a la unidad de datos con la que trabaja Kafka (que si lo vemos desde el punto de vista de las aplicaciones sería la unidad de comunicaciones entre aplicaciones).
- Cada mensaje esta compuesto por: clave, valor y un timestamp.
- Con componentes extra como el Schema Registry o Apache Avro podemos aplicarle a los mensajes un esquema.

Brokers

Brokers

- Instancia o un nodo/servidor de Kafka
- Es el mediador de las comunicaciones para la entrega de los mensajes y se comunica con los demás brokers utilizando Apache Zookeeper.
- Sólo un broker del clúster podrá tener la partición leader de un topic.
- A nivel de configuración tiene un identificador único para distinguirlo del resto de los brokers.

Brokers

- Cualquier broker en un cluster de kafka puede actuar como bootstrap server (es el broker de Kafka que se proporciona para obtener los metadatos iniciales, como pueden ser: topics, sus particiones, los brokers líderes para esas particiones, etc.).

Replicación

Replicación

- La replicación es una copia de una partición en otro broker, esto permite tolerancia a fallos, evitar perdidas de datos).
- La información es replicada entre todos los nodos del cluster (todos los brokers) pero no en todos.
- El factor de replicación es el número de copias de datos sobre múltiples brokers.
- El factor de replicación permite a Kafka ser tolerante a fallos y asegura que no hay pérdida de datos.

Replicación

- El número máximo de replicas es el número máximo de brokers en el cluster.
- El valor del factor de replicación debe ser mayor a 1 siempre (entre 2 o 3).
- Si un nodo esta caído la información puede ser entregada por cualquiera de los otros nodos.
- Mientras mas alto el factor de replicación podemos tener mayor disponibilidad (obviamente a mayor costo).

Criterios de retención

Criterios de retención

- Para la retención por lo general se configuran una serie de criterios y el incumplimiento de alguno de estos criterios hace que los mensajes “expiren” y sean eliminados.
- Por defecto el periodo de retención de los mensajes es de 7 días.
- Otro criterio de retención utilizado es el de tamaño ocupado.

Producers

Productores

- Es un cliente (aplicación) que se encarga de escribir mensajes en los topics.
- También se denomina "publicador", "publisher", "editor" o "escritor".
- A los productores se les pasa una lista de los servidores de Bootstrap (en esta lista aparece el nombre del host y el puerto de al menos un servidor de Bootstrap).

Productores

- La escritura del dato por parte del productor incluye 5 pasos:
 - Serialización
 - Particionado
 - Compresión
 - Acumulación
 - Agrupación

Consumidores

Consumidores

- Los consumidores son los clientes conectados, suscritos a los topics para consumir los mensajes.
- También se denomina "subscriber", "suscriptor" o "lector".
- Puede estar suscrito a uno o más topics.
- Cada consumidor tiene asociado un grupo de consumidores.
- Kafka garantiza que Cada mensaje sólo es leído por un consumidor de cada grupo.

Grupo de consumidores

Grupo de consumidores

- Es una agrupación de uno o más consumidores que trabajan para leer de un topic en base al cumplimiento de algún objetivo.
- Cada grupo se considera un único “consumidor”.
- Todos los consumidores de un grupo mantienen un balance de carga similar.

Garantías de entrega

Garantías de entrega

Tiene tres maneras de garantizar la entrega del mensaje:

- Como máximo una vez (at most once): garantiza que no hay duplicados pero puede perderse.
- Al menos una vez (at least once): garantiza que el mensaje siempre se entregara.
- Exactamente una vez (exactly once): garantiza que no hay duplicados ni hay perdidas.

Transacciones

Transacciones

- A nivel de transacciones permite realizar escrituras atómicas a varios topics y particiones en donde todos los mensajes incluidos en la transacción serán escritos con éxito o ninguno lo será.

Ejemplos

1. Shell
2. .NET

Complementos

1. Schema Registry
2. Kafka Connect
3. Kafka Strams
4. KSQL
5. Mirror maker
6. UI

Schema Registry

Schema Registry

- Registrar esquemas de datos en formato AVRO o JSON en un Repositorio centralizado.
- También asegura que los datos se insertan con un esquema concreto (para cumplir la especificación/contrato entre productores y consumidores)
- Tienen un API REST para consultar el histórico y versiones de los esquemas
- Tiene una UI.

Kafka Connect

Kafka Connect

- Es un framework que proporciona la capacidad de conectar kafka con sistemas externos para mover datos hacia o desde nuestro cluster. Se puede usar un modelo pull por ejemplo JDBC o un modulo PUSH como las herramientas CDC (Change Data Capture detecta los cambios en una serie de entornos a medida que se producen). Tiene un API que permite a los desarrolladores ejecutar productores y consumidores que conectan los topics de kafka con aplicaciones o sistemas externos que ya existen.

Kafka Strams

Kafka Strams

- Es una librería open source que nos permite construir aplicaciones de procesamientos de flujos de datos usando Apache Kafka como sistema de almacenamiento de entrada y salida de datos.

KSQL

KSQL

- Es un lenguaje similar al SQL para trabajar con streaming.
- Es una abstracción sobre la API de Kafka Streams que consume datos estructurados como AVRO y JSON de los topics de kafka, tiene baja latencia y permite guardar el estado del sistema

Mirror maker

Mirror maker

- Es una herramienta (script) diseñada para transferir datos de uno o más tópicos entre un cluster origen y otro destino.
- Esta es herramienta bastante útil para hacer replicado de datos sin que esto suponga un gran esfuerzo más que el coste de la configuración inicial.

Interfaz de Usuario

Interfaz de Usuario

- [AKHQ](#)
- [Kowl](#)
- [Kafdrop](#)
- [UI for Apache Kafka](#)
- [Lenses](#)
- [CMAK](#)
- [Confluent CC](#)
- [Conduktor](#)

Casos de uso

1. Monitorización de sistemas, redes o aplicaciones
2. Sistemas de recomendación
3. Notificaciones en tiempo real
4. IoT

Preguntas?



Thank you

@plainconcepts

www.plainconcepts.com