

ENMILOCALFUNCIONA

THOUGHTS, STORIES AND IDEAS.



Aprendiendo Apache Kafka (Parte 1): Introducción

Publicado por **Víctor Madrid** el 27 September 2018

[Arquitectura de Soluciones](#) [Kafka](#)

El objetivo que se persigue con esta **serie de artículos** es "aprender" a utilizar de la forma más práctica posible la plataforma **Apache Kafka** con todas sus características y particularidades. La estrategia que se seguirá en ellos será enseñar la teoría para tener una idea de cuál sería el funcionamiento, realizar la instalación, ir haciendo diferentes configuraciones para cubrir la mayoría de las casuísticas y mostrar ejemplos que las cumplan.

No te preocupes si al principio no entiendes nada, es normal, algunas veces aprender es como un buen vino "hay que dejarlo reposar un tiempo para sacar el máximo".

Por eso, a medida que se vaya avanzando en los artículos se irán poniendo en práctica conceptos y/o parte de ellos con ejemplos cuya complejidad se irá incrementando en cada caso, así que lo dicho, no te preocupes si al principio parece que las cosas no tienen sentido porque al final lo tendrá.



Trataré que la serie de artículos sea lo más práctica posible (*pero esto no siempre se cumple*, todo cuando hablamos de cosas que tienen cierta complejidad a la hora de entenderse y cuando acuerdas

Subscribe

interviene la tecnología... ya me entendéis... jeje), así que prometo que éste será el artículo más "pesado" de la serie... es muy teórico y en algunos puntos complicadillo.

En este **primer artículo** se enseñarán los **conceptos/elementos** desde la perspectiva de la **teoría** de Apache Kafka basada en la **definición, características, configuración y funcionamiento** de cada una de las partes que lo compone, de esta forma, conseguiremos centralizar casi toda la **teoría** en un **único punto** para que si en algún momento surgen las dudas se puedan encontrar rápidamente. De hecho, en algunos artículos se volverá a pedir que vuelvas a "echar un vistazo" a esta parte.

Este artículo está dividido en 5 partes:

- **1. Introducción General:** Se hará una breve introducción sobre ámbito de aparición de este tipo de tecnologías y sus áreas de aplicación.
- **2. Patrón "Publish / Subscribe Messaging":** Explicación sobre este enfoque arquitectónico.
- **3. Apache Kafka:** Información "general" asociada y casos de uso de la plataforma.
- **4. Conceptos Básicos:** Lista de elementos que será necesario conocer para entender bien la plataforma
- **5. Conclusiones:** Opinión sobre lo que se ha explicado.

Empezamos a darle "caña" a Kafka... ¿estás preparado?

1. Introducción General

Hoy en día cualquier empresa que se considere "seria" se mueve y evoluciona gracias a los datos que maneja.

¿A qué nos referimos cuando hablamos de datos?

Los **datos** (*un dato es la mínima unidad semántica*) no son más que un **conjunto discreto de valores** sobre las "cosas" con las que trabajamos y que deberían de tener una **serie de propiedades** concretas:

- *Compleitud*
- *Precisión*
- *Consistencia (no lleven a contradicciones)*
- *Unicidad*
- *Temporalidad*

Por ejemplo: el identificador utilizado para diferenciar un elemento del resto, la cantidad de unidades de un elemento, la fecha de entrega, el estado de un elemento, el nº de teléfono, etc.

Ayuda: Con el cumplimiento de la mayor parte de las anteriores propiedades por cada dato se consigue que su calidad sea mejor.

Hay que tener en cuenta que **los datos inicialmente y por si solos no aportan mayor valor a la toma de decisiones**, simplemente muestran el aspecto al que representan (por ejemplo: el valor "7" sabemos que representa un nº entero, pero no conocemos nada más), es decir, no están preparados para explicar el

porqué de las cosas, pero el **análisis e interpretación** de sus valores en conjunto y conociendo el contexto/utilidad **aporta** lo que se conoce como **información** (*conjunto de datos procesados que tienen un significado y que son de utilidad a la hora de tomar decisiones*) y como bien se sabe actualmente **la información es poder**.

Hay que diferenciar: datos, información y conocimiento:

- **Datos** = Conjunto discreto de valores
- **Información** = **Datos** + Contexto + Utilidad
- **Conocimiento** = **Información** + Experiencia + Otros

Por ejemplo: el valor "7" se puede entender de diversas maneras: como la cantidad de elementos de algo, como un valor de una tipología, como un identificador de un objeto, etc.

Por ejemplo: una fecha sobre un elemento se puede corresponder a su "creación", su "modificación", su "devolución", etc. Depende del ámbito que representen, así podríamos ver que si varios elementos tiene muchas fechas de "devolución" es que algo no está funcionando para ese elemento pero para ello se requiere ubicarlo en un contexto de tiempo.

Por lo tanto, **disponer** de los **datos correctos**, con la **calidad requerida**, en los **sitios y momentos necesarios** pasa a ser una **prioridad crítica** para **cualquier compañía** que quiera resultar competitiva hoy en día.

Hay una cita muy interesante que dice que "**Cada byte de un dato tiene una historia muy interesante que contar**" y al final va a tener toda la razón ;-).

Nota: todo esto es lo que se denomina **pipeline del manejo de datos** que se englobaría dentro del **gobierno de datos**.

¿Qué ocurre al utilizar cualquier aplicación?

Cualquier persona que haya utilizado alguna **aplicación** en su vida sabe que usarla implica **crear datos** de alguna manera (voluntariamente o involuntariamente) en algún momento (como poco a lo mejor guarda el usuario y password para que se pueda volver a entrar :-)).

Algunas de las **formas de almacenar la información** más típicas son:

- Generar un fichero
- Generar una entrada/registro en base de datos

Estos datos **aportan el valor necesario a la funcionalidad** que cubren y se encuentran **disponibles para su explotación** de forma independiente o bien son datos consumidos por otro conjunto de aplicaciones: accediendo a la misma base de datos, lectura de ficheros, creación de ficheros de intercambio "intermedios", peticiones REST, etc.

Con esto podemos ser conscientes de que constantemente a la hora de utilizar aplicaciones se están produciendo las siguientes **actividades**:

- **Obtener la información desde diferentes orígenes** (suelen ser muy variados y pueden requerir de diferentes mecanismos).

- **Manipular** la información para adaptarla a las diferentes necesidades (a veces se requiere la ejecución de operativas para dejar la información en el formato y/o con el contenido requerido para poder utilizarlo).
- **Analizar** la información con diferentes procedimientos y criterios.
- **Generar** la información de salida en base a lo anterior (para ayudar a esa toma de decisiones, aportar valor o bien servir de entrada para otra aplicación).
- **Mover** la información entre localizaciones o bien entre aplicaciones (a veces utilizando la carga masiva entre sistemas, otras veces).

De lo que se puede extraer que: "casi es tan importante la forma de obtener el dato, su manipulación y la forma que tenemos de moverlo (intercambiarlo) para su consumo."

2. Patrón "Publish / Subscribe Messaging"

Antes de "meternos en harina" conviene aclarar este concepto para no generar ninguna duda a la hora entender cómo funciona Apache Kafka por dentro.

¿Qué es?

Patrón de uso dentro de la tipología de arquitectura de "Cola de mensajes", utilizado para la comunicación entre aplicaciones.

Por lo tanto, se engloba más en el movimiento de información, aunque también puede cumplir aspectos de preparación o modificación.

También se denomina "publish / subscribe", "publicador / suscriptor" o "productor / consumidor".

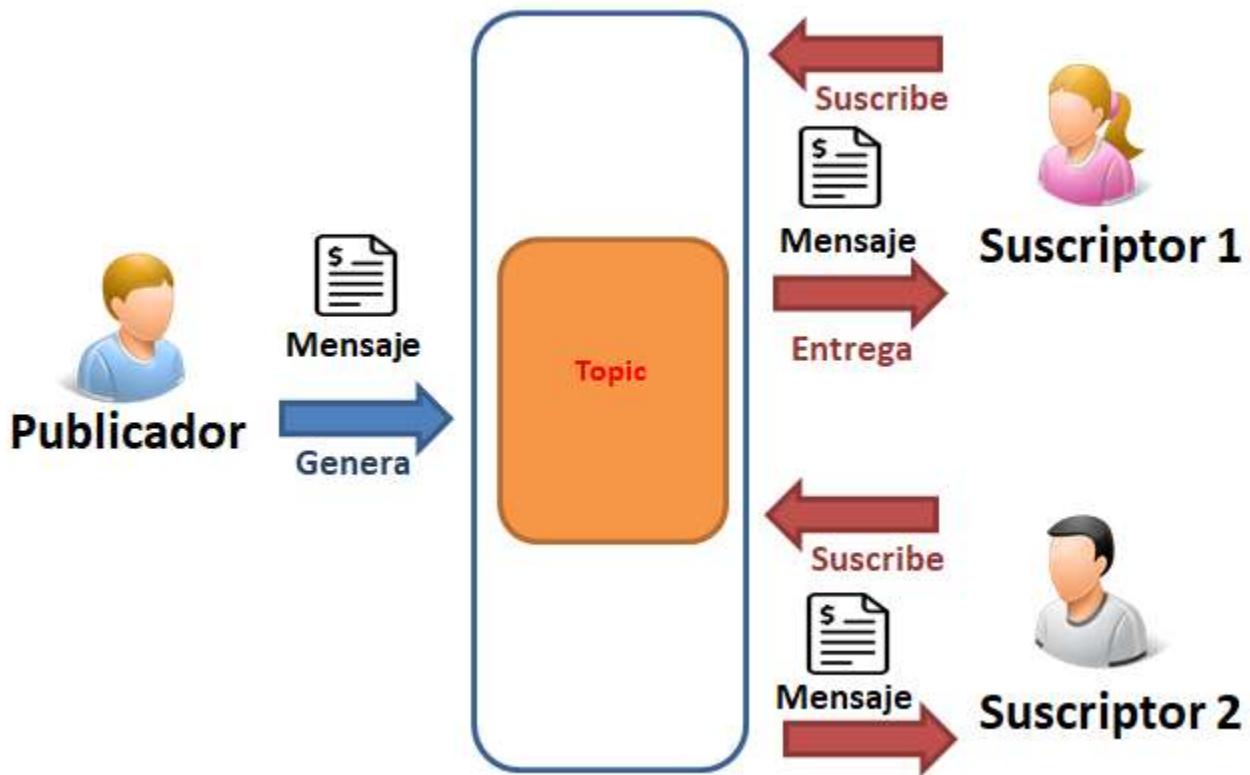
¿Cómo funciona?

Existe un elemento "**publisher**" (publicador / remitente / emisor / productor) que al **generar un dato** (message / mensaje / record / registro) **no lo dirige** o referencia específicamente a un "**subscriber**" (receptor / subscriptor / suscriptor) en concreto, es decir, no lo envía de forma directa a la "dirección" del subscriber.

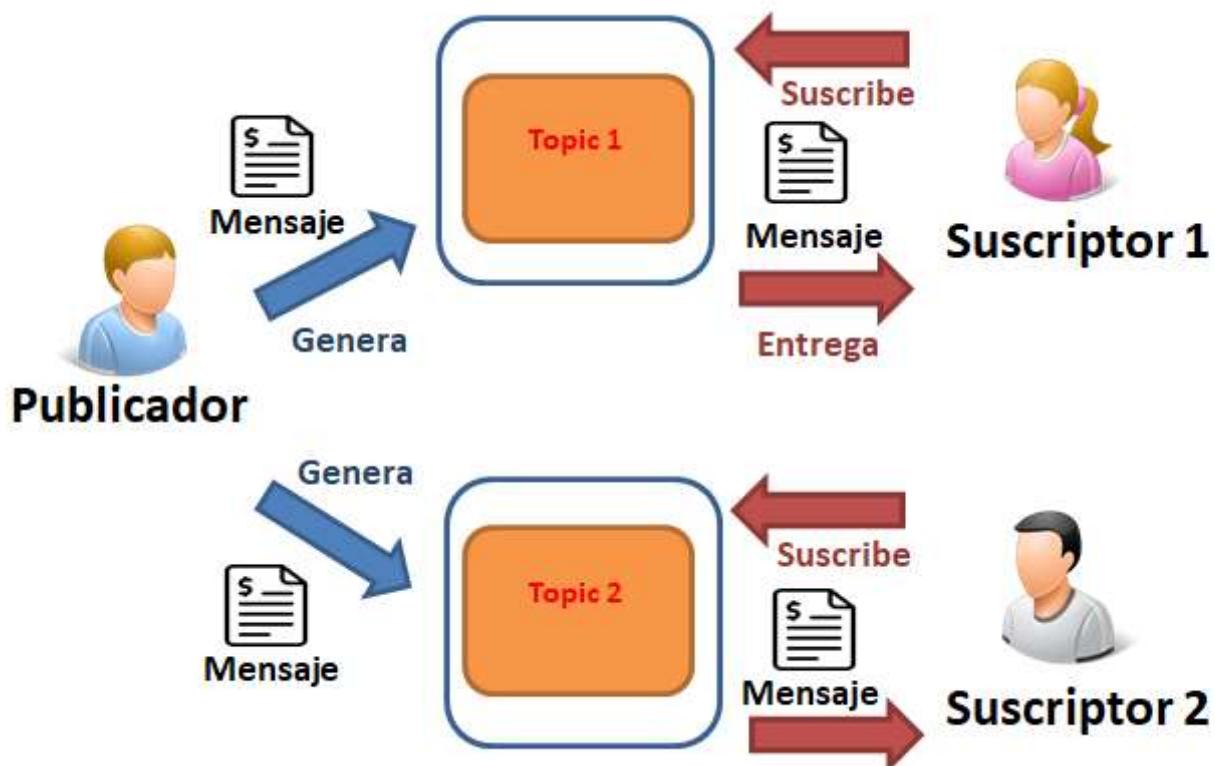
Para ello se **dispone de listas de temas/topics** publicados específicos y un **conjunto de suscriptores**, el productor trata de clasificar el mensaje en base a una tipología, lo pone en la lista de un tema específico y el receptor se suscribe a la listas para recibir ese tipo de mensajes.

Para ayudar en su comprensión se van a mostrar varios diagramas conceptuales:

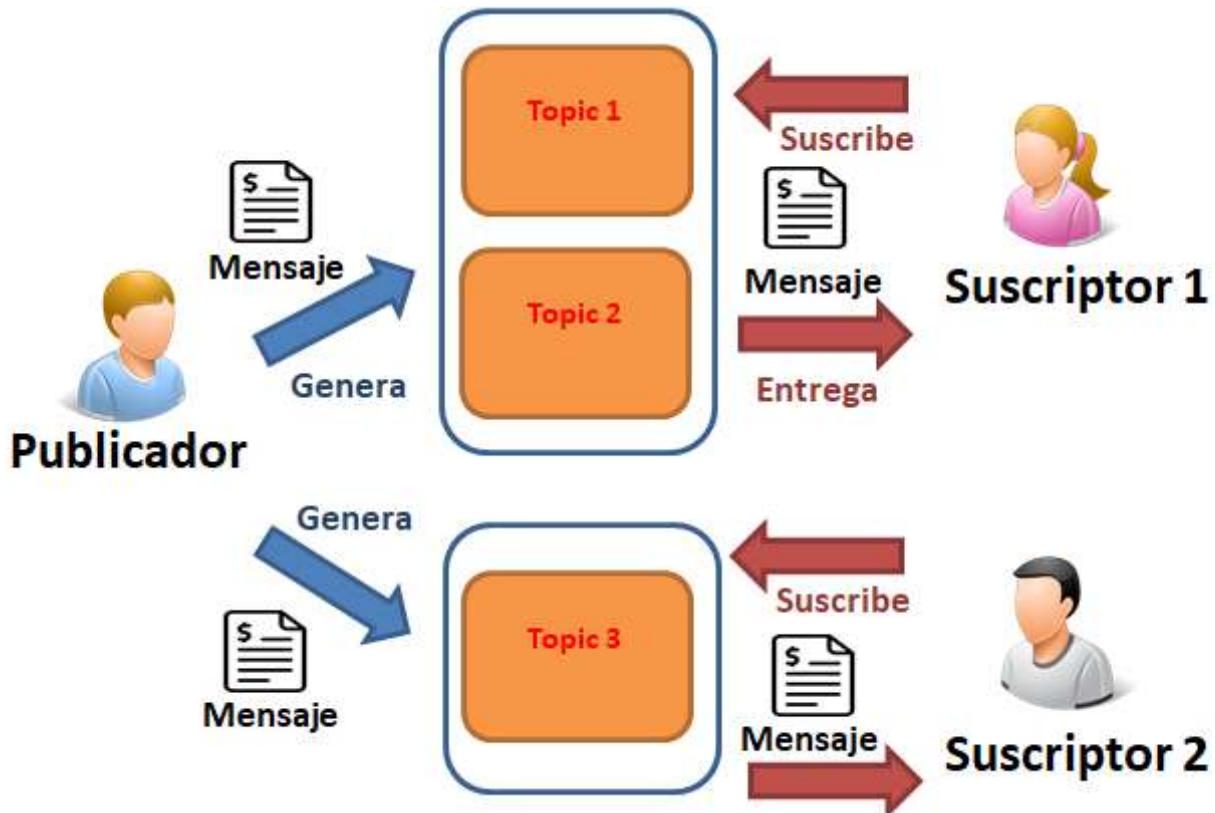
- **Diagrama Conceptual:** "Un único tema/topic y dos suscriptores"



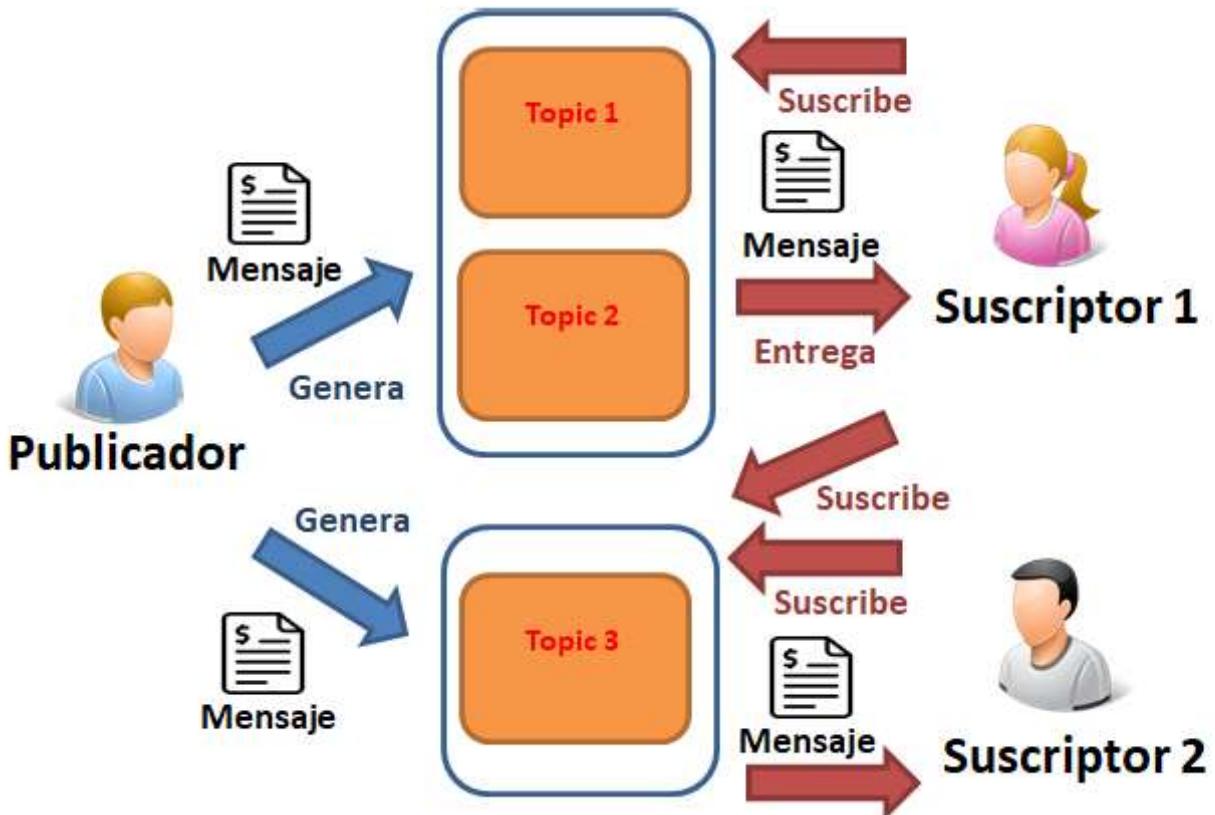
- **Diagrama Conceptual:** "Dos temas/topics y un suscriptor por tema/topic"



- **Diagrama Conceptual:** "Varios temas/topics y un suscriptor con varios temas/topics específicos"



- **Diagrama Conceptual:** "Varios temas/topics y uno de ellos es compartido por varios suscriptores"



Detalles

- Este tipo de comunicación se utiliza sobre todo para la **comunicación asíncrona**.

Permite **diferentes tipos de configuración**:

- **Tradicional:** Cada suscriptor está asociado a uno o varios topic en concreto. Existen muchas variaciones:

- Cada suscriptor está escuchando 1 topic propio.
- Cada suscriptor está escuchando X topics independientes.
- Cada suscriptor está escuchando X topics independientes y Y topics compartido.
- **Grupos de consumo:** Los suscriptores se pueden agrupar por grupo, este grupo está escuchando un topic y sólo un miembro del grupo tendrá la capacidad de atender el mensaje.
- **Radio Difusión:** Todos los suscriptores que están escuchando el topic reciben el mensaje (cada suscriptor es responsable de interpretar el mensaje de forma independiente).
- Requiere de otra pieza que sirve de intermediario (broker) donde se publican los topics.
- Proporciona un sistema centralizado que permite la publicación de tipos genéricos de datos y que puede evolucionar con el tiempo.

Consideraciones

A la hora de realizar una valoración de este tipo de enfoques habría que tener en cuenta los siguientes aspectos:

- **Precisión (Correctness):** Debería de definir alguno o una combinación de las siguientes estrategias (con/sin duplicados, con/sin orden y con/sin perdida de datos) que se encargan de establecer garantías de entrega y de ordenación
- Estrategias de **entrega**:
 - "**como máximo una vez**" (**at most once**): Sólo se envía una vez, garantiza que no hay duplicados pero puede perderse.
 - "**al menos una vez**" (**at least once**): Garantiza que no hay perdida.
 - "**exactamente una vez**" (**exactly once**): Garantiza que no hay duplicados y que tampoco hay perdidas.
- Estrategias de **ordenación**:
 - **No hay ordenación (No ordering)**: No importa el orden de recepción y por lo tanto se puede incrementar el rendimiento.
 - **Ordenación por partición (Partitioned ordering)**: Asegurar un orden por partición y por lo tanto tiene un coste extra.
 - **Ordenación Global (Global Order)**: Se requiere un orden en los datos por lo tanto necesita un mayor coste extra de los recursos y posibles implicaciones en el rendimiento.
- **Disponibilidad (Availability)**: Capacidad para estar el mayor tiempo posible trabajando o activo. En este punto hay que tener en cuenta los mantenimientos y la aparición de errores (detección + reparación).
- **Transaccionalidad (Transaction)**: Capacidad para agrupar acciones o elementos en unidades atómicas. Es decir, "O se ejecutan todas como una única operación o bien no se ejecuta ninguna".
- **Escalabilidad (Scalability)**: Capacidad de un elemento para evolucionar en un aspecto con el objetivo de poder dar soporte a una mayor cantidad de acciones o elementos -> Pueden existir diferentes dimensiones: mensajes, topics, productores o consumidores

- El objetivo es manejar la cantidad de tráfico.
- Suele entrar en conflicto con la eficiencia (filtrado complejo, enrutado complicado, etc. suele ser no escalable).
- **Rendimiento (Throughput)**: Capacidad relacionada con la medida de la eficiencia en lo referente a la cantidad (nº de bytes) de elementos por unidad de tiempo con los que puede trabajar -> En algunos casos también se denomina ancho de banda (bandwidth).
- Se engloba dentro del concepto de "eficiencia"
- **Latencia (Latency)**: Capacidad relacionada con la medida de la eficiencia basada en el pipeline requerido para su procesamiento, es decir, se suele considerar el tiempo requerido para procesar un elemento -> En algunos casos también se denomina tiempo de respuesta.
- Se engloba dentro del concepto de "eficiencia".
- Suele ser inversamente proporcional respecto al rendimiento
- **Desacoplamiento (Decoupling)**: Capacidad que hace referencia a diferentes aspectos:
- **Entidad (Entity)**: Los clientes (productor y consumidores) no necesitan conocerse entre si
- **Tiempo (Time)**: Los clientes (productor y consumidores) no necesitan participar activamente
- **Sincronización (Synchronization)**: Si los hilos de ejecución o los clientes requieren algún tipo de bloqueo síncrono
- **Enrutamiento Lógico (Routing logic)**: Capacidad por la que un dato que sale de un productor acaba en un consumidor concreto. Sería aplicar la lógica de negocio del caso de uso y se puede enfocar en base a:
- **Tema (Topic)**: Los datos se clasifican en temas y los suscriptores sólo reciben datos relacionados con esos temas. Una variante de este tema sería la especialización de los temas en base al tipo de objeto.
- **Contenidos (Content)**: Los datos se clasifican en base a uno o varias propiedades y los suscriptores pueden establecer filtros o restricciones específicas sobre un grupo de datos.

3. Apache Kafka

<https://kafka.apache.org/>

La plataforma de Apache Kafka se trata de un sistema de mensajes "**publish / subscribe**" Open Source basado en una arquitectura P2P (arquitectura Peer to Peer).

Se entiende como un **commit log** que es "**log (registro) de commit (confirmación) distribuido**".

Un "commit log" tiene las siguientes **características**:

- *Estructura de datos ordenada y persistente.*
- *Es el core de Kafka.*
- *Sólo permite añadir por el final (anexar).*
- *No se pueden modificar ni borrar registros.*

- Se pueden invalidar valores según ciertos criterios.
- Proporciona un procesamiento determinista.

Mejora la forma de trabajo con los datos en las aplicaciones a la hora de realizar comunicaciones y/o procesamientos con ellos.

Para ello proporciona un enfoque de **bus de mensajes** (con productores y consumidores que usan listas para trabajar con mensajes).

Destaca por:

- Plataforma para la nueva generación de aplicaciones distribuidas.
- Tiene mucha popularidad en la actualidad (muchas grandes empresas lo tienen incorporado en sus arquitecturas: Netflix, Microsoft, Bancos, aseguradoras, empresas ticketing, etc).
- Tiene una orientación hacia el mundo "Big Data" aunque también destaca en otros ámbitos como la comunicación de aplicaciones, la ejecución de procesos batch, etc.
- Está escrito en Java y Scala.
- Fue desarrollado en sus inicios como herramienta de "apoyo" en LinkedIn pero posteriormente se hizo Open Source (Apache Community).
- El nombre no tiene ninguna referencia "oculta" hacia el escritor más que la propia elección del nombre.
- Pertenece al ámbito "Distributed Messaging Queue".
- Alternativa a JMS, AMQP y RabbitMQ cuando se maneja volúmenes importantes de datos/información y se requiere un gran capacidad de respuesta -> mejora sus características más importantes.
- Facilita el trabajo con otras tecnologías como: Flume / Flafka, Spark Streaming, Storm, HBase, Flink y Spark para ingerir, analizar y procesar datos en tiempo real.
- Aplicaciones online: Apache Solr, Logstar.
- Procesamiento streams: SAMZA, Storm, Flink, Wallaroo.
- Procesamiento offline: Hadoop.
- Dispone de un API de "Producer": Facilita que una aplicación publique una secuencia de mensajes en uno o más topics de diferentes formas.
- Dispone de un API de "Consumer": Facilita que una aplicación pueda suscribirse a uno o más topics y así poder procesar la secuencia de mensajes.
- Dispone de un API de "Streams": Facilita procesar un flujo consumiendo un flujo de entrada de uno o más topics y produciendo un flujo para uno o más topics de salida.
- Dispone de un API de "Connector": Facilita implementar/ejecutar productores o consumidores reutilizables con el objetivo de conectar topics con aplicaciones o sistemas de datos existentes.

3.1. Características

- **Alto rendimiento:** Dispone de la capacidad para trabajar con poca latencia debido a que depende del núcleo del SO (Sistema Operativo):
- Cumplimiento del "**zero copy**" (evita copiar buffers en memoria)
- Mantiene bajo el heap size de la JVM
- **Escritura secuencial en disco** (escritura en el commit log inmutable) -> tipo O(1)
- Hace uso de registros inmutables -> Escribe al sistema de archivos (registro de temas de Kafka) secuencialmente
- **No realiza acceso aleatorio a los datos**
- Trabaja con registros de datos en **chunks** / fragmentos / lotes de datos
- Facilita la compresión y reduce la latencia
- *Evita la pérdida de datos*
- *"Suele" ser a nivel de rendimiento*
- *Aspectos como la red podrían afectar*
- *Se aconseja monitorizar las métricas*
- **Distribuido:** Se divide en varios nodos para su ejecución, los cuales trabajan de forma conjunta al trabajar dentro de un clúster
- Para el usuario final se ve como un único nodo
- Este enfoque proporciona: escalado horizontal y tolerancia a fallos
- **Escalado horizontal:** Se puede escalar en muchos nodos en relativo poco tiempo
- Facilita agregar un nuevo nodo sin tiempos de inactividad
- No suele tener "límites" en la cantidad de nodos
- Uso de sharding
- Se puede fragmentar las listas / temas / topics en particiones y repartirlas por los nodos
- **Durabilidad / Persistencia:** Los mensajes son persistidos en el sistema de fichero y son replicados entre los clusters (se pueden establecer criterios para invalidar valores pero si se persiste consigue que sean duraderos y confiables)
- **Tolerancia a fallos:** No tienen un único punto de fallo bloqueante (SPoF) al replicar las particiones (de un topic log) en múltiples servers
- Cuanto mayor es la tolerancia menor es el rendimiento
- **Replicación:** La información será replicada entre los nodos del clúster
- Consigue disponer de "Alta Disponibilidad" y de "Recuperación inmediata"
- **Políglota:** La comunicación se realiza usando un protocolo de comunicaciones basado TCP por lo que se pueden implementar mediante diferentes lenguajes
- **Retro compatible:** Trata de asegurar el correcto funcionamiento con versiones anteriores
- **Integrabilidad:** Dispone de un ecosistema que proporciona un proxy REST (No está proporcionado por el proyecto Apache) que facilita la integración mediante HTTP y JSON.

- Además otra de sus características es que resulta muy útil para la comunicación y la integración entre componentes, aplicaciones, etc. (cuando manejan gran cantidad de datos)
- **Soporte a Avro/Schema** (No está proporcionado por el proyecto Apache): Facilita la gestión de los esquemas (Confluent Schyema Registry)
- **Seguridad** (opcional): Cubre varios aspectos que están relacionados:
 - Soporte para ciertas medidas de seguridad como: SSL, SASL (PLAINTEXT y SCRAM), GSSAPI (Kerberos), ACL, etc
 - Autenticación de conexiones de brokers con Zookeeper (fichero JAAS, modificar la propiedad "zookeeper.set.acl")
 - Crifrado de datos entre brokers y clientes /herramientas / productores / consumidores (con soporte SSL)
 - El cifrado es sólo en vuelo
 - La seguridad puede tener un coste de rendimiento
 - Fácil instalación y configuración
 - Proporciona KSQL (lenguaje similar al SQL para trabajar con streaming)
 - La publicación y la suscripción de los mensajes se puede realizar gracias a la serialización y deserialización
 - Se ejecuta en una máquina virtual de Java (JVM)
 - Se ve afectado por elementos como: la recolección de basura, tamaño del heap, métricas (como Daemon, Peak y Live Thread Count), etc.
 - Tratar de evitar que los hilos puedan sobrecargar la memoria del servidor
 - Se deberá de controlar los tamaños
 - Proporciona un sistema de herramientas de base para ayudar a tareas de sistemas o bien de gestión
 - Ejecutar una clase de script: kafka-run-class
 - Gestionar las operaciones sobre los topics
 - Migrar un broker de una entre versiones
 - Verificar la posición del offset en un consumidor
 - Replicar particiones entre nodos
 - Replicar los clusters (Mirror Maker)
 - ...

Algunas de sus desventajas son:

- **Extra de Mantenimiento:** se dificulta enormemente los mantenimientos al implementar sus componentes de forma heterogénea y descontrolada
- **Mal enfoque:** un mal diseño o planteamiento sin patrones, sin arquitectura etc., puede afectar de forma directa al rendimiento

- **Selección "*/comodín":** No soporta enviar el mensaje a todos los topics, siempre se tiene que indicar un nombre específico del topic
- **Monitorización:** las herramientas para monitorización no son muy "alla"
- Ejemplos: Yahoo Kafka-Manager, Burrow, KafkaOffsetMonitor, etc.
- **Ajustes de mensajes:** Si un mensaje requiere de ajustes puede provocar un empeoramiento del rendimiento al poder afectar a las llamadas internas al sistema
- **Rendimiento:**
 - Por regla general los mensajes no se comprimen, pero en algunas ocasiones puede ser interesante hacerlo.
 - Cuando un mensaje se descomprime ocupa memoria de un broker
 - Cuando se incrementa el número de colas el rendimiento se puede resentir

3.2. Casos de uso o aplicaciones

Al principio de la introducción se hablaba de la importancia de controlar y gestionar de forma correcta los pipeline del manejo de datos. En este apartado se va a hablar sobre los usos de esta plataforma y donde puede ayudar a mejorar la forma de trabajo:

- *Uso en arquitecturas de streaming de datos en tiempo real*
- *Envío/intercambio/transformación de datos entre aplicaciones*
- *Manejar flujos de datos en tiempo real (por ejemplo: procesamiento de streams, agregación de registros, ingesta de datos en Spark, etc.)*
- *Recopilar grandes datos (por ejemplo: logs, changelogs, tracking actividad de una web, monitoreo de métricas, seguimiento de clicks, etc), lo que facilita el aprendizaje y mejora*
- *Procesamiento de eventos complejos (CEP)*
- *Funcionamiento como una "cache" persistente*
- *Proporciona durabilidad en microservicios in-memory*
- *Proporciona eventos a CEP (Complex Event Streaming System) y a sistemas IoT/IFTTT* Etc.*

En definitiva, casi todo lo que tenga que ver con: Messaging (mensajería -> broker de mensajes), Metrics (métricas -> información de monitorización operacional) y Event Sourcing (gestión de eventos)

Aclaración

Todos conocemos el registro de commit/confirmación de un sistema de archivos o bien de una base de datos, su objetivo es el de proporcionar un registro duradero y persistido de todas las transacciones que se han realizado.

Una utilidad muy interesante de disponer de ese registro es la capacidad de poder *construir el estado de un sistema de forma consistente*

Símil: Sería como disponer de la foto de un lugar que ha sido modificado con el tiempo y gracias a esa foto podemos tratar de dejarlo como lo que aparece en la foto" -> Para volver a empezar desde ese punto

En resumen Apache Kafka realiza la misma acción con los datos, así los puede persistir de forma duradera y facilita su lectura de forma ordenada y determinista con el extra de que al proporcionar distribución obtiene mecanismos de escalabilidad y de recuperación frente a fallos.

4. Conceptos Básicos

En este apartado se van a listar algunos de los conceptos y/o elementos correspondientes al framework de Apache Kafka:

- Zookeper
- Broker
- Clúster
- Mensaje
- Esquema
- Topic / Tema
- Partición
- Offset
- Kafka Connect / Conector
- Kafka Streams
- Productor
- Consumidor
- Grupo de Consumidores

Todos estos elementos serán explicados en detalle en el siguiente artículo

5. Conclusiones

En este artículo se ha podido ver a nivel teórico donde se encuentra la potencia de esta plataforma (bus de mensajes con ciertas características y con un enfoque orientado hacia Big data) y porque es tan popular a la hora de trabajar de ciertas empresas que manejan/consumen gran cantidad de datos .

Pocos serían capaces de decir que "NO" a una plataforma "única" distribuida de transmisión de eventos/mensajes en tiempo real con almacenamiento duradero y que proporciona de base un alto rendimiento (capaz de manejar billones de peticiones al día), tolerancia a fallos, disponibilidad y escalabilidad.

De hecho, se "rumorea" que esta plataforma la utilizan una tercera parte de las empresas que componen Fortune 500 y muchas otras empresa. De forma que se está convirtiendo casi en un estándar de facto si se manejan ciertos volúmenes de datos y que posiblemente tendrá una evolución muy grande en los próximos tiempos.

¿Nos animamos a utilizarla?

Autor

VÍCTOR MADRID

Líder Técnico de la Comunidad de Arquitectura de Soluciones en atSistemas. Aprendiz de mucho y maestro de nada. Técnico, artista y polifacético a partes iguales ;-)

COMPARTE



ALSO ON EN MI LOCAL FUNCIONA

Acelerando los desarrollos con ...

hace 2 años • 1 comentario

En este artículo se va a mostrar cómo poder disponer de un ...

Las tripas de SonarQube: Métricas ...

hace 2 años • 1 comentario

En este post vamos a verle las tripas a SonarQube y a entender por qué y cómo ...

Phaser 3: Mi primer juego HTML5

hace 2 años • 3 comentarios

Continúanós hablándoos de cómo desarrollar un videojuego completo con ...

Da
Pc

hac
En
có
de

 Acceder ▾

29 Comentarios

G

Únete a la conversación...

[INICIAR SESIÓN CON](#)[O REGISTRARSE CON DISQUS](#)

Nombre

14

[Comparte](#)[Mejores](#)[Más nuevos](#)[Más antiguos](#)**S****santi (atSistemas)**

hace 4 años

Gran articulo crack!!

Esperando leer la 2º parte

1 0 [Responder](#) • [Comparte](#) >**J****johann** → santi (atSistemas)

hace un año

Para proyecto de 3 meses conocimientos en apache kafka
escribi a jmaceo@manapro.com0 0 [Responder](#) • [Comparte](#) >**D****David Borrego Gutierrez**

hace 4 años

Muy buen artículo Víctor, deseando ver la segunda parte. Un abrazo!

1 0 [Responder](#) • [Comparte](#) >**J****johann** → David Borrego Gutierrez

hace un año

Buscando perfiles junior para proyecto en telefonica con apache kafka

0 0 [Responder](#) • [Comparte](#) >**Earningram**

hace 8 meses edited

Excelente toda la explicación y también me gustaría que expliques más sobre bases de Datos NoSQL (No relacionales) como Mongo entre otros

0 0 [Responder](#) • [Comparte](#) >**Juan David Nicholls**

hace un año

Está genial la introducción, gracias por compartir! <3

0 0 Responder • Comparte >

Condiciones de Uso

J

johann

hace un año

Para proyecto de 3 meses buscando perfiles junior en bigdata, conocimientos en apache kafka
escribi a jmaceo@manapro.com

Powered by atSistemas

0 0 Responder • Comparte >

M

Marc Estrader



hace 2 años

Muy bien explicado, Asi da gusto. Muchas gracias !

0 0 Responder • Comparte >

F

furra



hace 2 años

que tipo de diagrama utilizarías para representar una coreografía con kafka

0 0 Responder • Comparte >

A

Armando



hace 3 años

Saludos, muy buena documentación, tengo una duda respecto a esto:

"Grupos de consumo: Los suscriptores se pueden agrupar por grupo, este grupo está escuchando un topic y sólo un miembro del grupo tendrá la capacidad de atender el mensaje."

Solo un miembro atiende el mensaje y que sucede con el resto de miembros? como se enteran del mensaje?

0 0 Responder • Comparte >

E

enver



hace 3 años

Muy claro y detallado, mas que animado para iniciar su uso.

0 0 Responder • Comparte >

C

Cristian Alvarez



hace 3 años

Muy claro, didáctico y explicado de forma sencilla. Gracias y felicitaciones.

0 0 Responder • Comparte >

J

Jefferson Mendoza



hace 3 años

Excelente articulo!

0 0 Responder • Comparte >

J**Jonatan**

hace 3 años

Felicitaciones! Muy buen post. Algunas bibliografias que nos puedas recomendar? Gracias

0 0 Responder • Comparte ▾

F**Freyder Otalvaro**

hace 3 años

excelente artículo. muchas gracias

0 0 Responder • Comparte ▾

E**Ezequiel Hernández**

hace 3 años

Tiene kafka la posibilidad de tomar un mensaje y luego volver a meterlo en la queue para que sea vuelto a tomar luego de X tiempo? donde el X tiempo sea un parámetro definido dentro de Kafka?

0 0 Responder • Comparte ▾

DA**Daniel Argueta**

hace 3 años

Hola amigo, esta muy interesante tu contenido.

Tendrías algún tipo de vídeo donde lo emplees usando java?

0 0 Responder • Comparte ▾

**Fernando Gutiérrez Paramio**

hace 4 años

Muy bueno el artículo. Creo que nos conocemos Víctor, je je je je. Soy uno que se fue a Colombia. Estaba buscando info buena para utilizar Kafka con Apache Flink y Kudu. Un abrazo compañero.

0 0 Responder • Comparte ▾

V**Víctor Madrid**

→ Fernando Gutiérrez Paramio

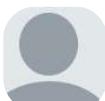
hace 4 años

Muy cierto...jejeje Claro que se quien eres :-)

Espero que te haya podido ayudar con el articulo

Otro abrazo para ti

0 0 Responder • Comparte ▾

**Noé Montes Alvarez**

hace 4 años

En primer lugar enhorabuena por el artículo. Estoy empezando a investigar estas tecnologías y este tutorial (y las siguientes partes) me parecen muy interesantes.

Mi duda es la siguiente. De las herramientas que propones para monitorización, cual dirías que es la mejor?

Muchas gracias. Un saludo!

0 0 Responder • Comparte >



Víctor Madrid

→ Noé Montes Alvarez

— ┆

hace 4 años

Buenas Noe,

Perdona por el retraso , pero no me llego ninguna notificación de tu comentario. Seguramente ya lo has resuelto porque ha pasado mucho tiempo :-) . Tendrías que decir cuales son las necesidades específicas para tu caso ya que como pasa siempre , cada herramienta tiene sus cosas buenas y malas para cada caso

0 0 Responder • Comparte >



Joskar Risquez

— ┆

hace 4 años

Enhorabuena Víctor por el articulo.

Para los que estamos aterrizando con el producto nos da un enfoque estupendo para entenderlo desde un punto de vista mas cercano!

Voy a por las siguientes partes ;-)

Un saludo y gracias por tu aporte!

0 0 Responder • Comparte >



Víctor Madrid

→ Joskar Risquez

— ┆

hace 4 años

Buenas Joskar,

Muchas gracias por tus palabras, tratare de seguir escribiendo para ayudar todo lo que pueda ;-)

Un saludo

0 0 Responder • Comparte >



Julian Nobsa

— ┆

hace 4 años

Tremendo artículo, me suscribo al tópico del próximo. ¿Cómo se podría configurar una alta disponibilidad? es decir, tener n instancias de kafka por si alguna falla.

0 0 Responder • Comparte >



Víctor Madrid

→ Julian Nobsa

— ┆

hace 4 años

Buenas Julian,

Muchas gracias :-)

La alta disponibilidad aparecerá cuando se trabaje con varios nodos,se configure la capacidad de replicación de un topic entre varios nodos y se vea que pasa cuando se "para" uno de los nodos con los que esta trabajando. Ya se mostrará ese punto en concreto en los siguientes artículos :-)

1 0 Responder • Comparte >



Be Ca Vas

— ┆

hace 4 años

Buen, artículo. Soy nuevo en kafka y me esta ayudando a entender mejor esta tecnología. Sin embargo, hay algo que no me queda muy claro, ¿kafka usa disco o memoria?

0 0 Responder • Comparte ▾



Víctor Madrid

→ Be Ca Vas



hace 4 años

Buenas Be Ca Vas

Muy buena pregunta , la respuesta sería que usa las dos cosas :-) . En primer lugar mantiene las cosas en memoria y luego dependerá de tu configuración a nivel de intervalo de flush (escritura en disco) el mover de memoria a disco

Espero haberte podido ayudar

Un saludo

0 0 Responder • Comparte ▾



Alejandro Sánchez



hace 4 años

Muy buen artículo Víctor!