Contenido

[APACHE KAFKA 2](#_Toc513046169)

[¿Qué es Apache kafka? 2](#_Toc513046170)

[Algunos conceptos 2](#_Toc513046171)

[Apache Kafka (Gráfico de servicios los cuales se pueden implementar) 2](#_Toc513046172)

[Un conjunto de herramientas para Streaming. 2](#_Toc513046173)

[Tema de investigación. 3](#_Toc513046174)

[Zookeeper 3](#_Toc513046175)

[kafka 4](#_Toc513046176)

[Manejador de eventos con apache Kafka. 5](#_Toc513046177)

# APACHE KAFKA

## ¿Qué es Apache kafka?

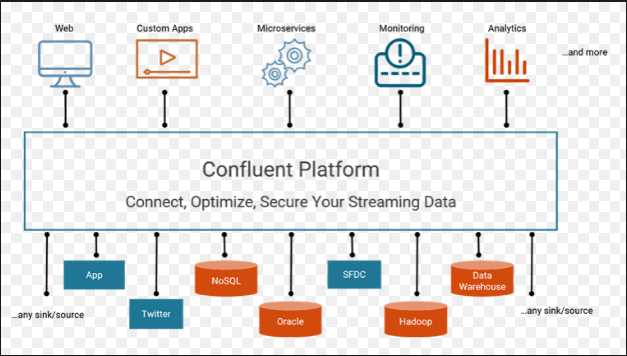
Apache Kafka es una plataforma de transmisión distribuida capaz de manejar billones de eventos al día. Inicialmente concebido como una cola de mensajes, Kafka se basa en una abstracción de un registro de compromiso distribuido. Desde que fue creada y abierta por LinkedIn en 2011, Kafka ha evolucionado rápidamente de la cola de mensajería a una plataforma de transmisión en toda regla.

Como plataforma de transmisión, Apache Kafka ofrece canalizaciones de publicación y suscripción de baja latencia, alto rendimiento y tolerante a fallas, y puede procesar flujos de eventos. Kafka proporciona respuestas fiables en milisegundos para admitir aplicaciones orientadas al cliente y conectar sistemas descendentes con datos en tiempo real.

Algunos conceptos:

* Kafka se ejecuta como un clúster en uno o más servidores que pueden abarcar múltiples centros de datos.
* El clúster de Kafka almacena secuencias de *registros* en categorías llamadas *temas*.
* Cada registro consta de una clave, un valor y una marca de tiempo.

## Apache Kafka (Gráfico de servicios los cuales se pueden implementar)



## Un conjunto de herramientas para Streaming.

Apache Kafka es una herramienta popular para los desarrolladores porque es fácil de utilizar y proporciona una potente plataforma de transmisión completa con 4 API:

* Productor.- Permite que una aplicación para publicar un flujo de registros a uno o más temas Kafka.
* Consumidor.- Permite a una aplicación suscribirse a uno o más temas y procesar la secuencia de registros que se les produce.
* Streams.- Permite que una aplicación actúe como un procesador de flujo, consumiendo un flujo de entrada de uno o más temas y produciendo un flujo de salida para uno o más temas de salida, transformando de manera efectiva los flujos de entrada a los flujos de salida.
* Connect.- Permite la construcción y en funcionamiento productores reutilizables o consumidores que conectan los temas Kafka a las aplicaciones existentes o sistemas de datos. Por ejemplo, un conector a una base de datos relacional puede capturar cada cambio en una tabla.

A menudo, los desarrolladores comenzarán con un caso de uso único. Esto podría ser utilizar Apache Kafka como un búfer de mensajes para proteger una base de datos heredada que no puede mantenerse al día con las cargas de trabajo de hoy, o usar la API Connect para mantener dicha base de datos sincronizada con un motor de indexación de búsqueda, para procesar los datos tal como llegan. la API de Streams para agregar agregaciones directamente a su aplicación.

En resumen, Apache Kafka y sus API hacen que la creación de aplicaciones basadas en datos y la administración de complejos sistemas de fondo sean simples. Kafka le da tranquilidad al saber que sus datos siempre son tolerantes a fallas, reproducibles y en tiempo real. Le ayudamos a crear rápidamente al proporcionar una plataforma de transmisión única para procesar, almacenar y conectar sus aplicaciones y sistemas con datos en tiempo real.

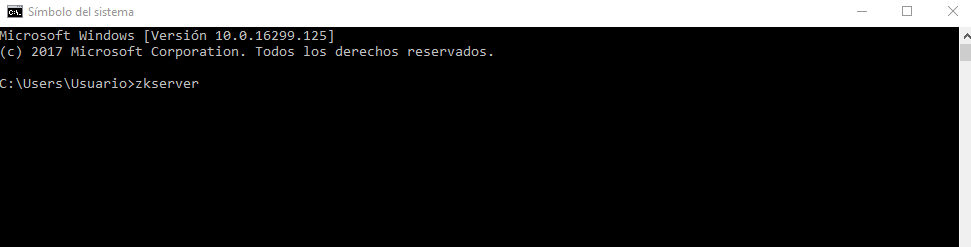
## Tema de investigación.

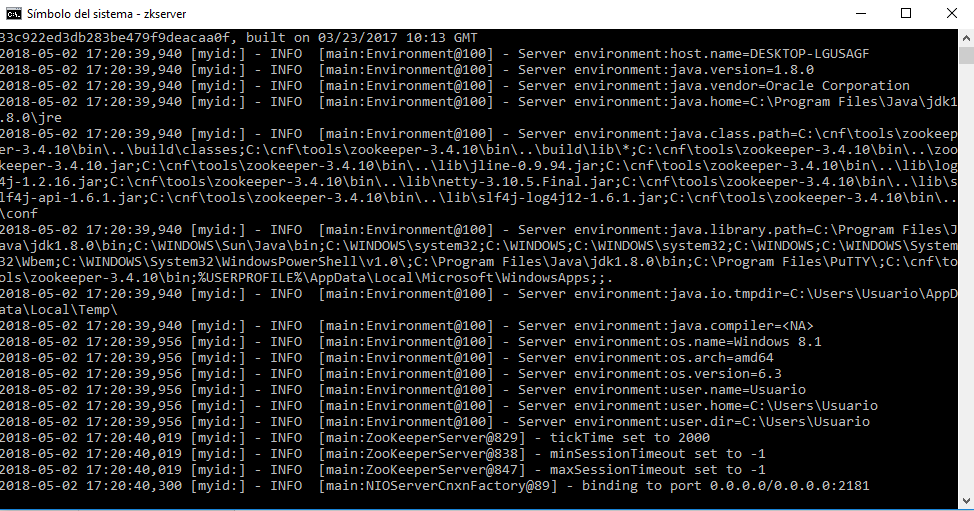
Crear un productor y un consumidor.

Requisitos:

1. Zookeeper
2. Apache Kafka

### Zookeeper

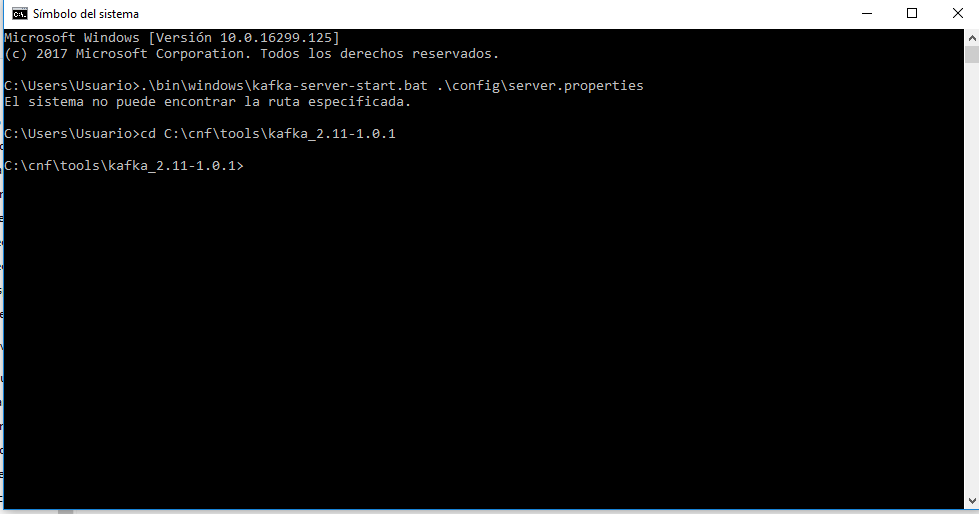


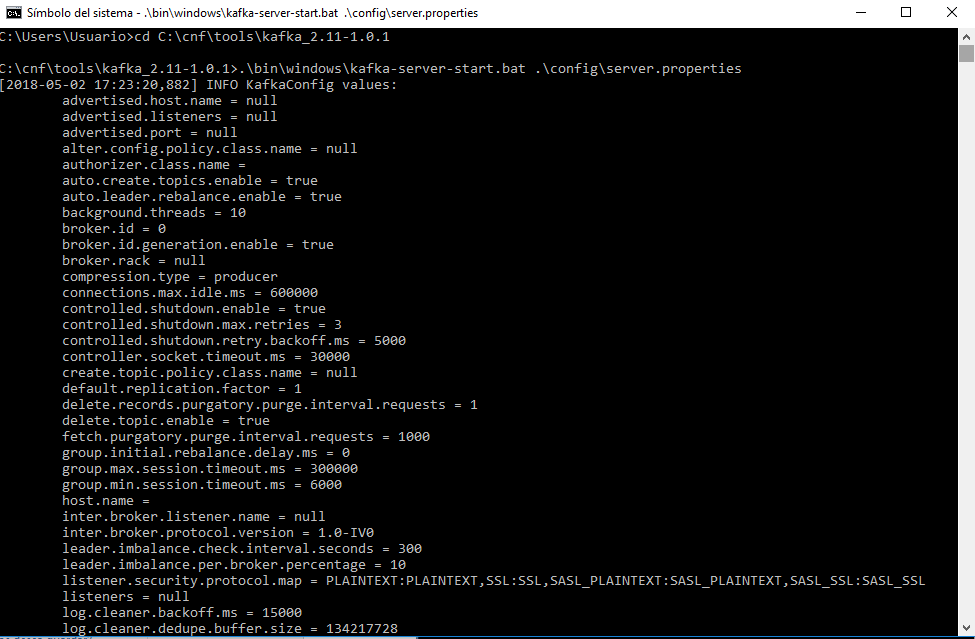


### kafka

Ir a esta ruta donde previamente se configuro (desde cmd) el apache Kafka y zokeeper: C:\cnf\tools\kafka\_2.11-1.0.1 y escribir el comando:

.\bin\windows\kafka-server-start.bat .\config\server.properties

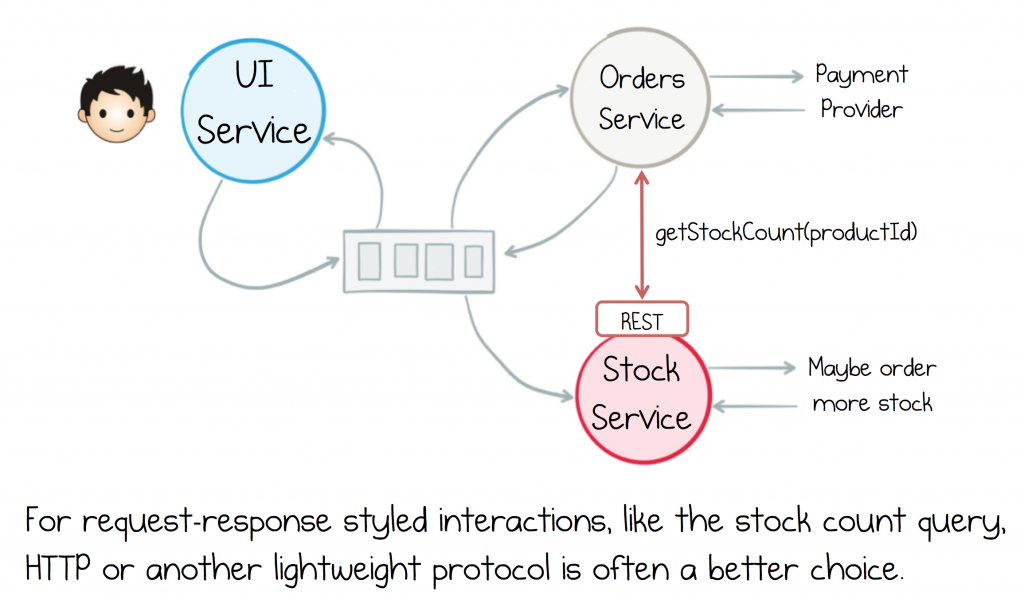




## Manejador de eventos con apache Kafka.

Kafka se utiliza en una amplia gama de casos de uso, pero como la mayoría de las tecnologías tiene su punto óptimo. Funciona muy bien para sistemas basados ​​en eventos. Sin embargo, a veces un estilo de solicitud y respuesta es más apropiado, entonces, ¿qué haces?

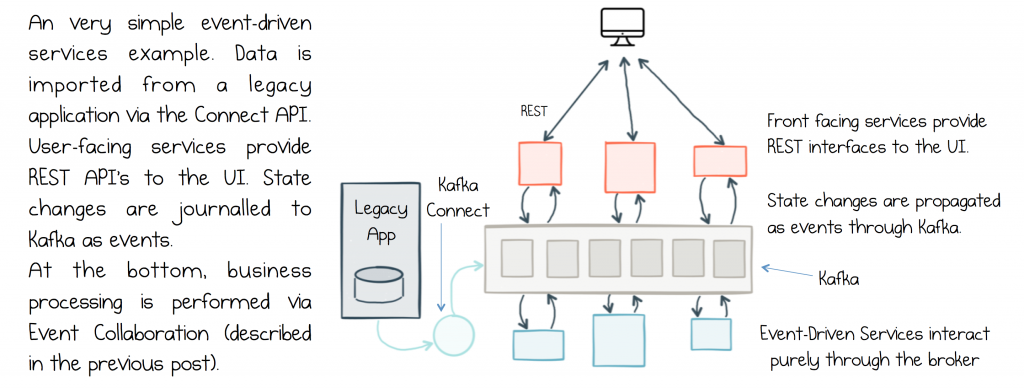
Digamos que necesita una forma para que el servicio de Pedidos consulte un recuento de stock del Stock Service. Puede sintetizar tal solicitud sobre cualquier canal asíncrono. Esto generalmente se hace creando dos temas, un tema de solicitud y un tema de respuesta.



Usar Kafka para comandos y consultas no es infrecuente, pero tampoco es su punto óptimo. Los temas de Kafka son más pesados ​​que, por ejemplo, HTTP. Si elige usar comandos o consultas a través de Kafka, utilice temas de partición única y reduzca el segmento y los tamaños de retención para mantener bajos los gastos generales. Alternativamente, mezcle en un protocolo sin estado como HTTP, en capas sobre la columna vertebral de los eventos. Los eventos son, después de todo, el conjunto de datos de su sistema. Las consultas son más como chit-chat efímero. Cubrimos patrones de diseño impulsados ​​por eventos en la última publicación.

**Construyendo un sistema simple basado en eventos**

Un patrón común para implementaciones de servicios más pequeños es mezclar protocolos: Kafka se usa para mover datos de un servicio a otro a través de una transferencia de estado llevada a cabo por eventos, activando servicios en acción o usándolos como un almacén de eventos. REST, o algún otro protocolo RPC, se usa para consultas, etc.



Pero debemos tener cuidado con este patrón a medida que nuestro sistema crece. Para arquitecturas más grandes, de tamaño corporativo, un patrón como el modelo de contexto agrupado (discutido en la última publicación) puede ser una mejor opción, ya que promueve el uso de transferencia de estado llevado por eventos a la vez que compartimenta las consultas remotas. Pero para aplicaciones de servicio más pequeñas, el patrón descrito anteriormente es un compromiso simple y efectivo.