

U.N. Sede Medellín

Una universidad con criterio nacional y presencia regional

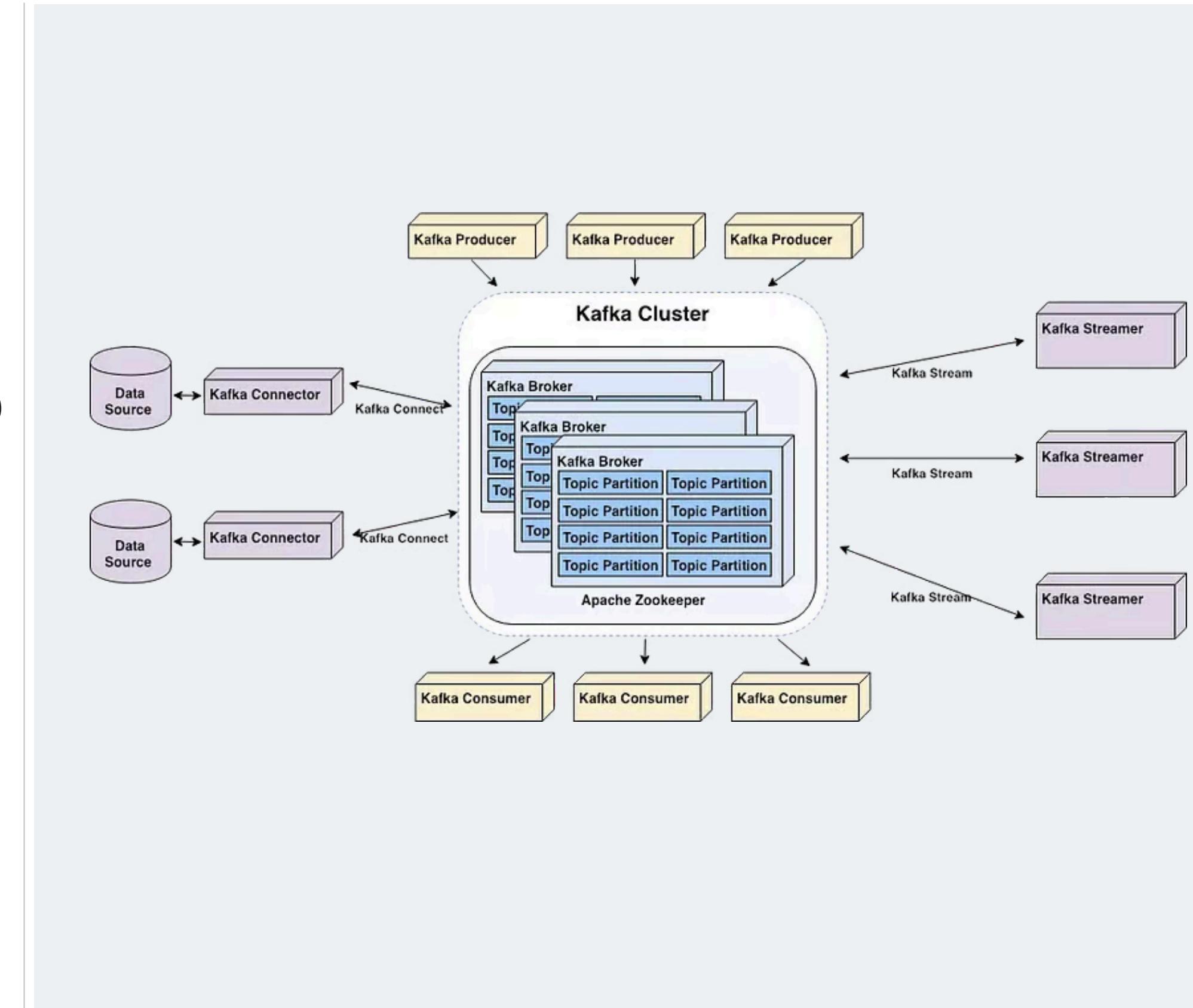




UNIVERSIDAD
NACIONAL
DE COLOMBIA

¿Qué es Apache Kafka?

Apache Kafka es una plataforma de streaming distribuida, de código abierto, diseñada para manejar grandes volúmenes de datos en tiempo real.



Tutorial (Opcional)

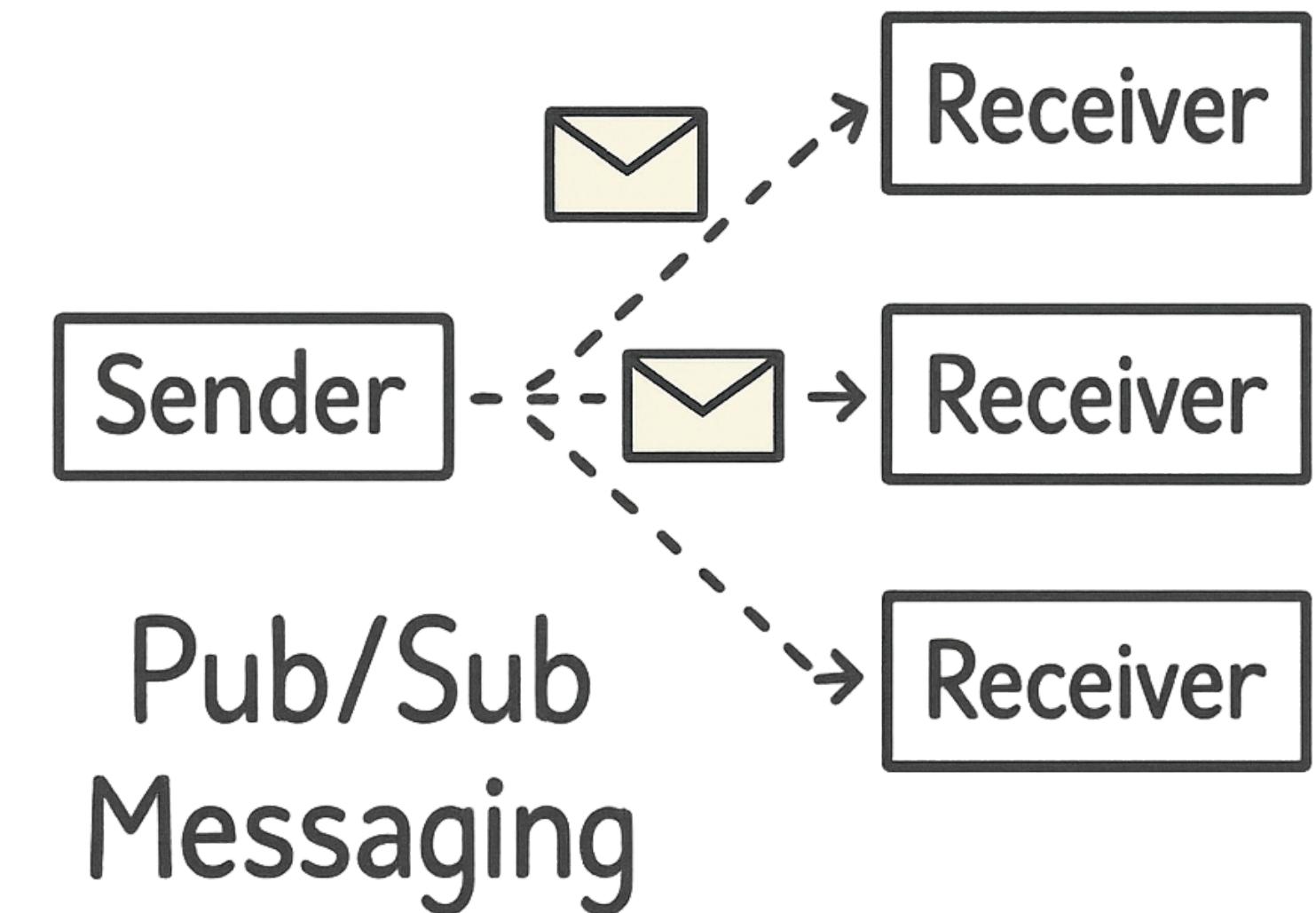


https://youtube.com/playlist?list=PLt1SlbA8guusxiHz9bveV-UHs_bIWFegeU&si=s_7GRAK3NvmeT0Mg

Sistemas de mensajería



Point-to-Point

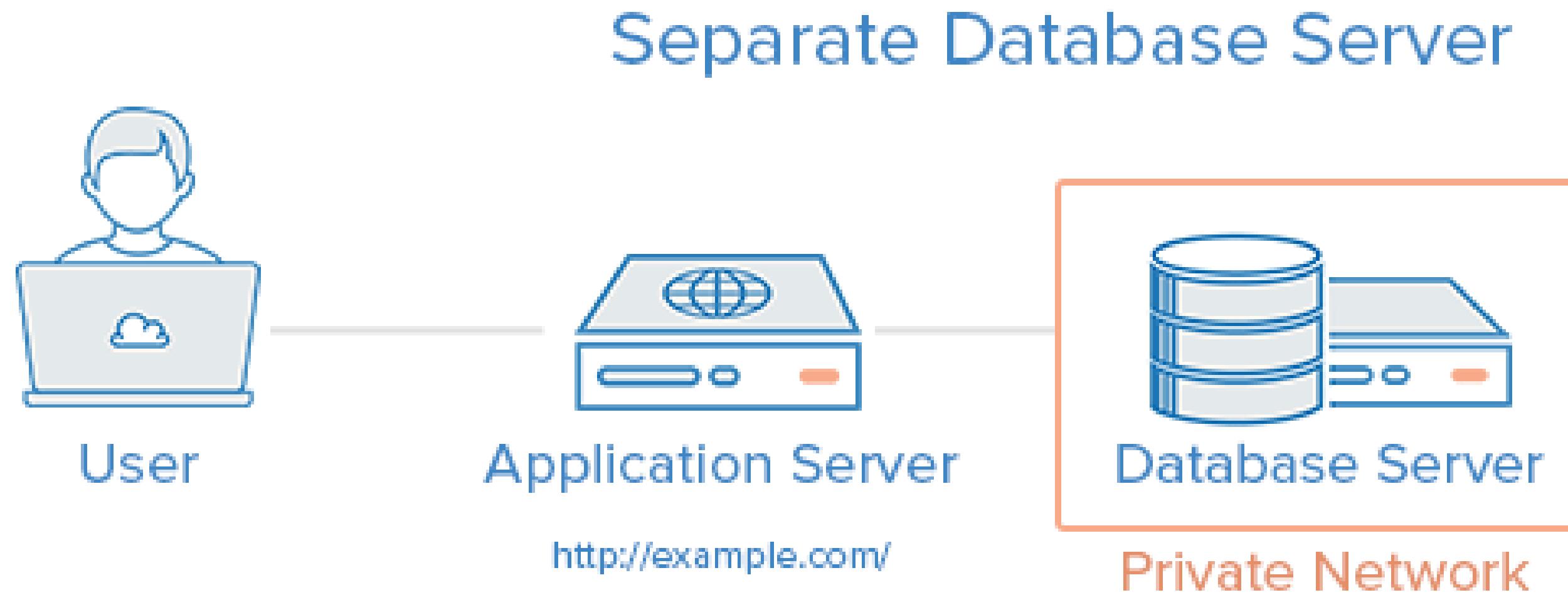


Pub/Sub
Messaging



¿Cómo empiezan las compañías ?

Manejo de datos simples



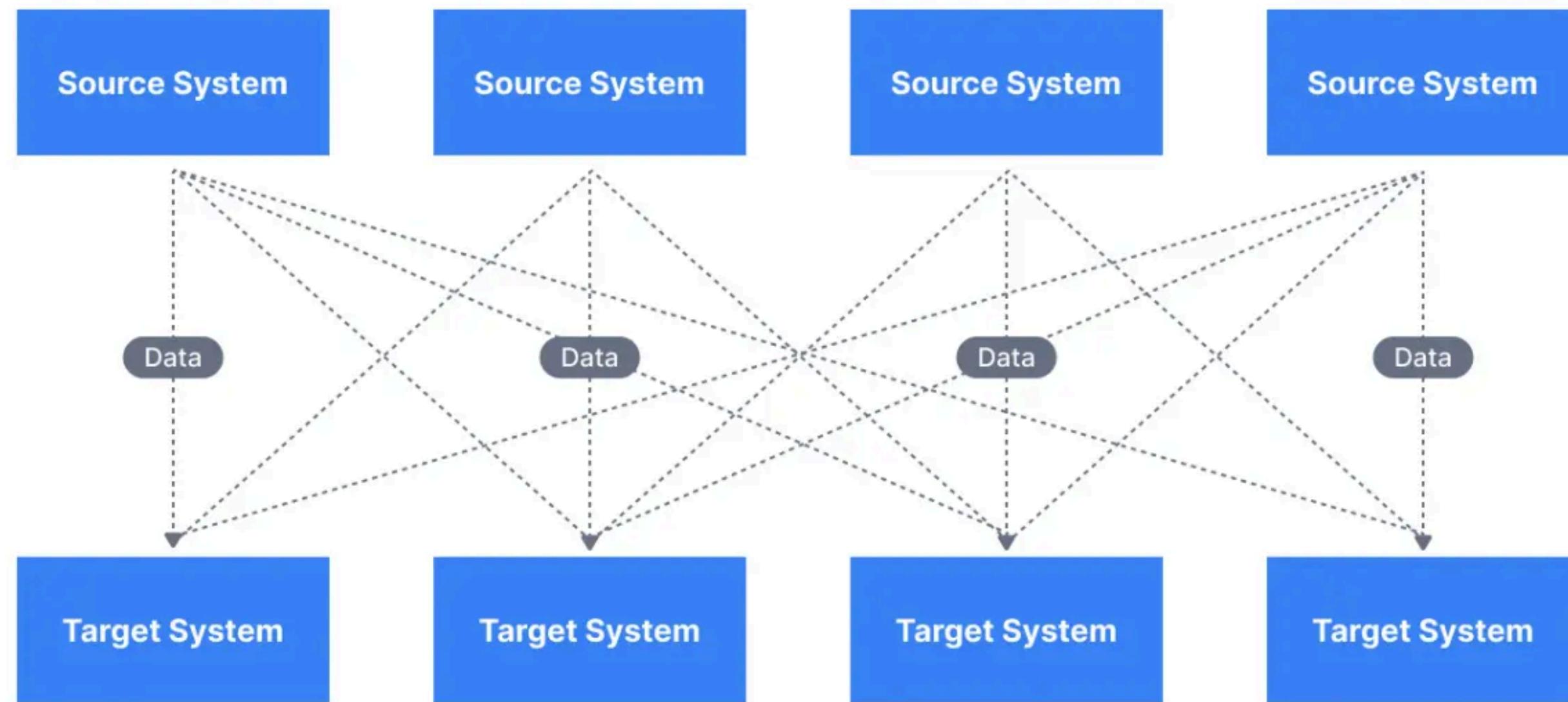
Kafka 101 – how does it achieve high throughput?

Messaging queues – like Apache Kafka, Apache ActiveMQ, and RabbitMQ, – are used in various...

© NeatCode / Dec 21, 2023

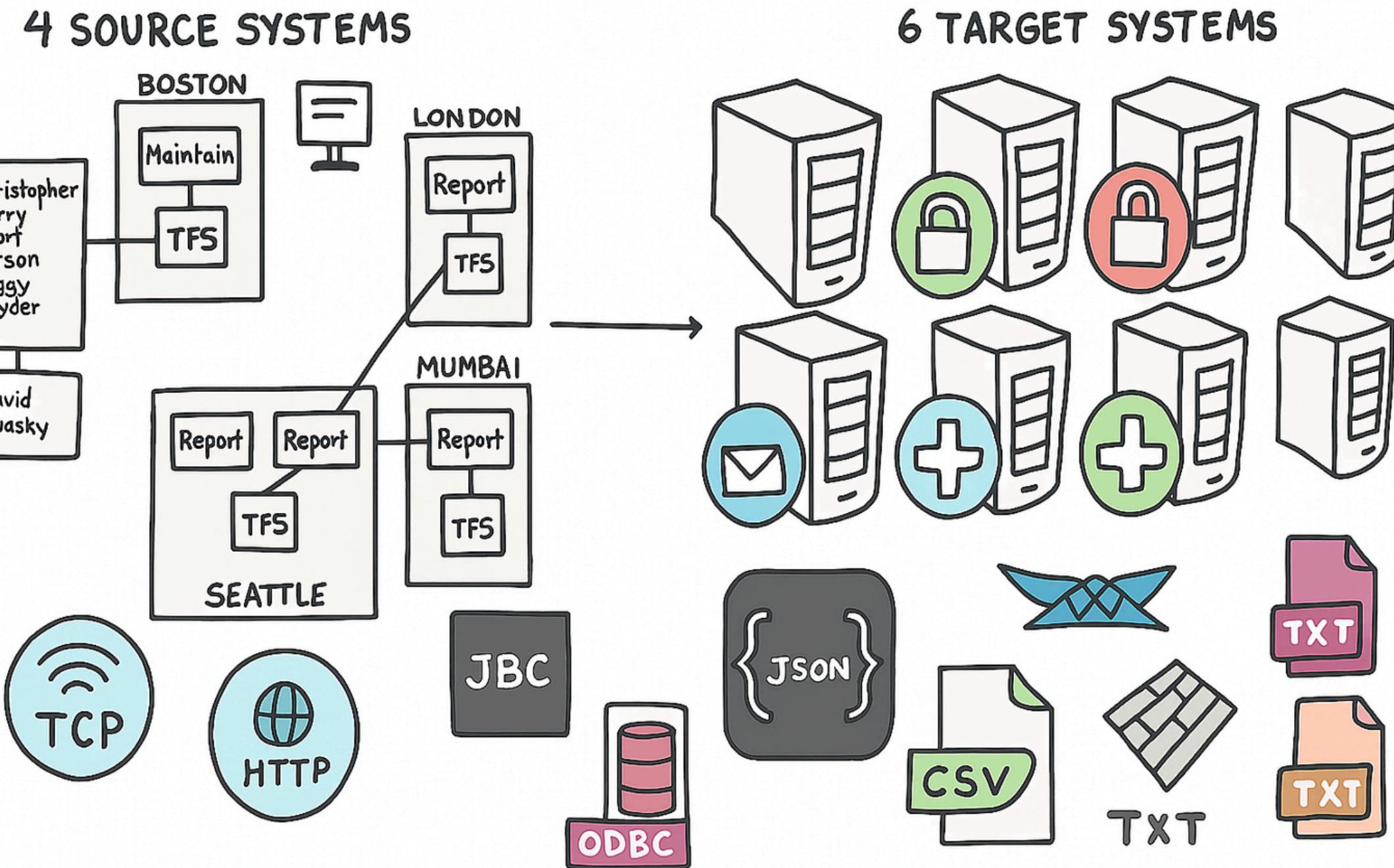
¿Después de un tiempo?

Una organización típica tiene múltiples fuentes de datos con formatos dispares. La integración de datos implica combinar datos de estas múltiples fuentes en una visión unificada del negocio.



¿Problemas de arquitectura?

Una organización típica tiene múltiples fuentes de datos con formatos dispares. La integración de datos implica combinar datos de estas múltiples fuentes en una visión unificada del negocio.



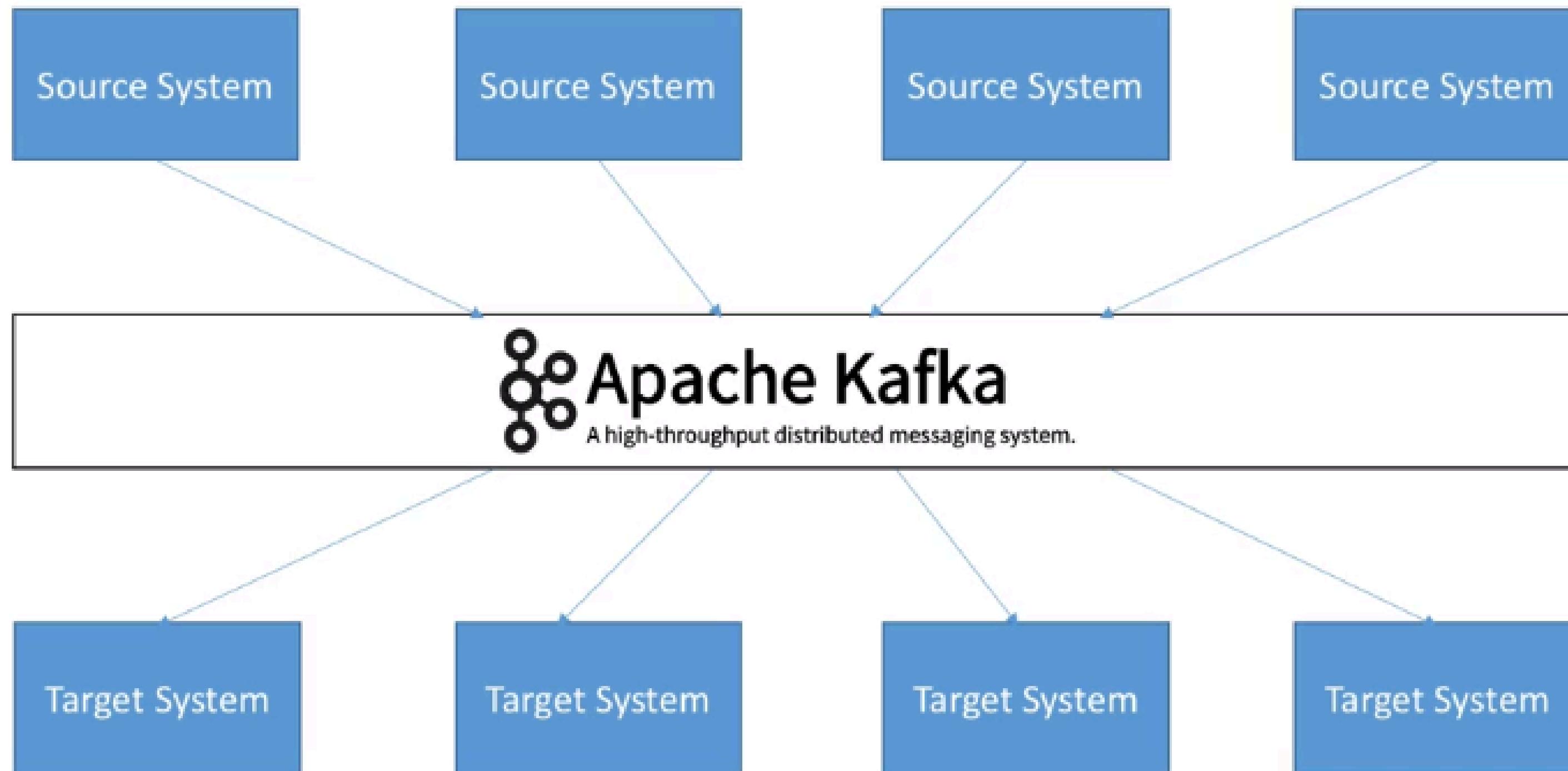
Kafka 101 – how does it achieve high throughput?

Messaging queues – like Apache Kafka, Apache ActiveMQ, and RabbitMQ, – are used in varik

© NeatCode / Dec 21, 2023

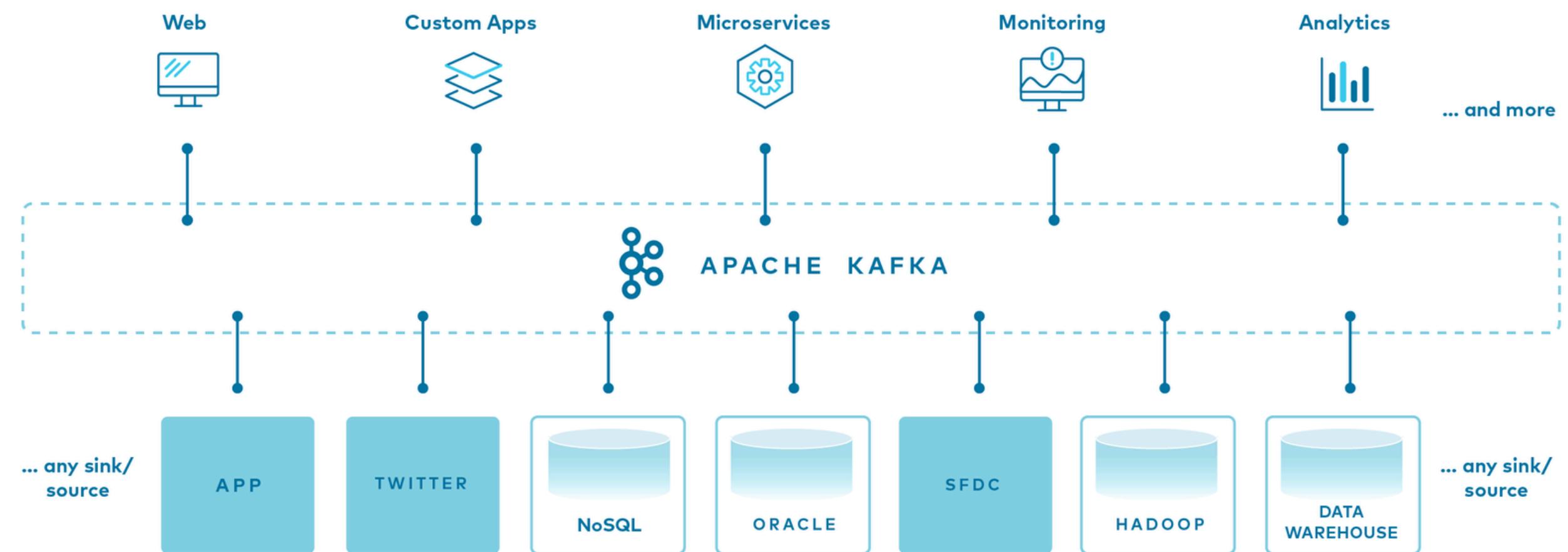
¿Por que Kafka?

La respuesta es simple, desacoplamiento y centralización de datos



¿Donde se usa?

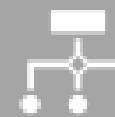
- Eventos de sitios web
- Datos de fijación de datos
- Transacciones financieras
- Acciones de usuarios
- Bases de datos
- Analítica
- Sistema de correos
- Auditoria
- Redes sociales
- Sistema de mensajería
- Seguimiento de actividades
- Integraciones con Big data



Características iniciales generales



Creación de pipelines de análisis de flujos de datos que obtiene los mensajes de forma confiable entre sistemas y aplicaciones



Es posible ejecutar Kafka en uno o varios servidores



Kafka almacena flujos de registros en categorías llamadas tópicos

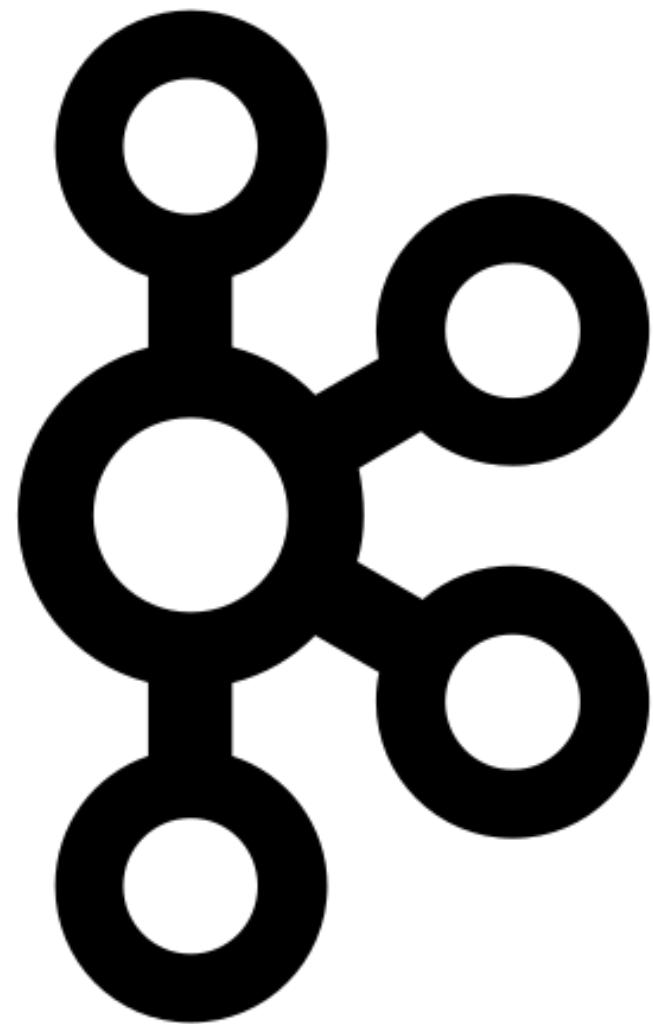


Cada registro contiene una llave, un valor y una marca de tiempo (timestamp)



Los mensajes de Kafka se conservan en el disco y se replican dentro del clúster para evitar la pérdida de datos.

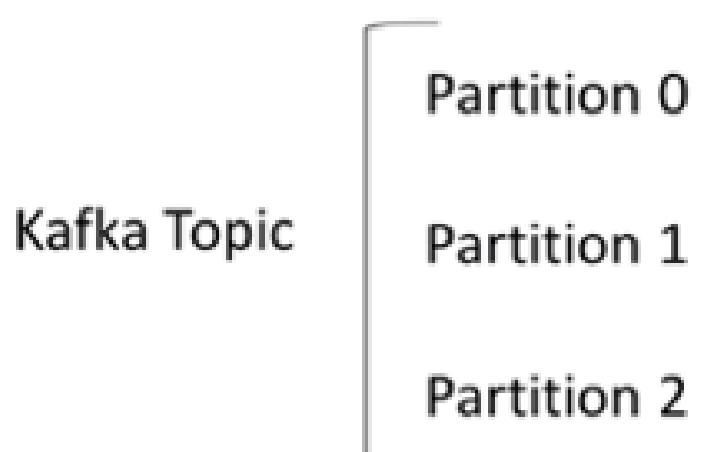
Definiciones



Topics

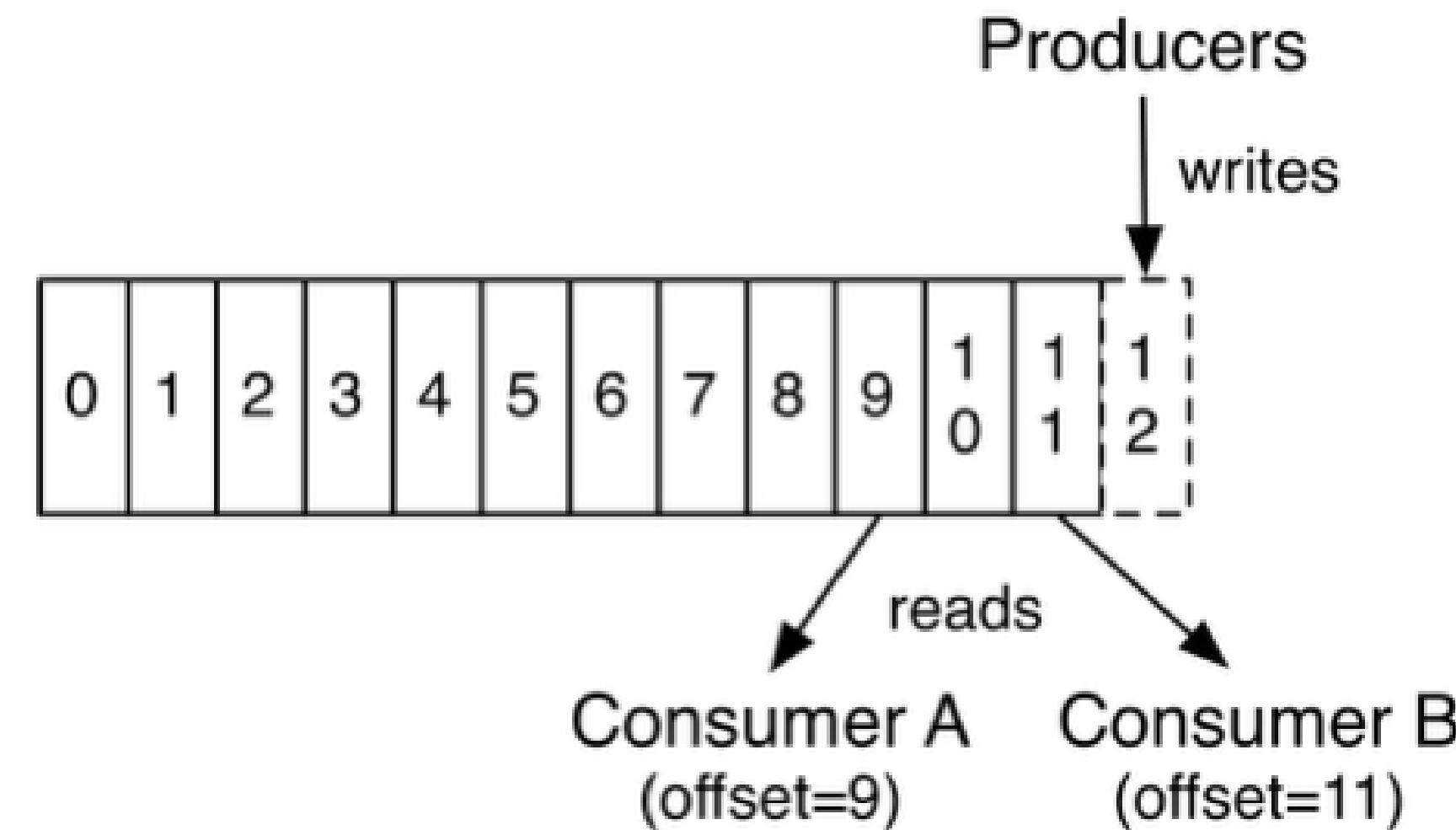
Es la base de todo en kafka, representa un flujo de datos en particular

- Similar a una base de datos, sin las restricciones
- Puedes tener cuantos tópicos tu deseas
- Un tópico es definido por su nombre
- Los topics están divididos en particiones
- Dentro de cada mensaje de una partición obtendremos un ID incremental llamado offset(compensaciones), el orden solo esta garantizado para cada partición



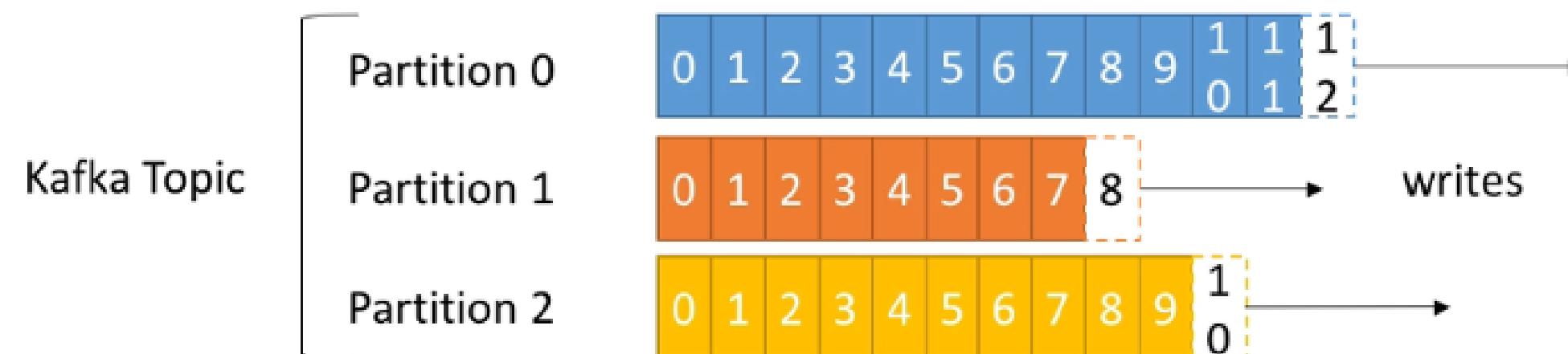
Offsets

(id-incremental) elemento de la partición o también conocida alias de la posición en la lista

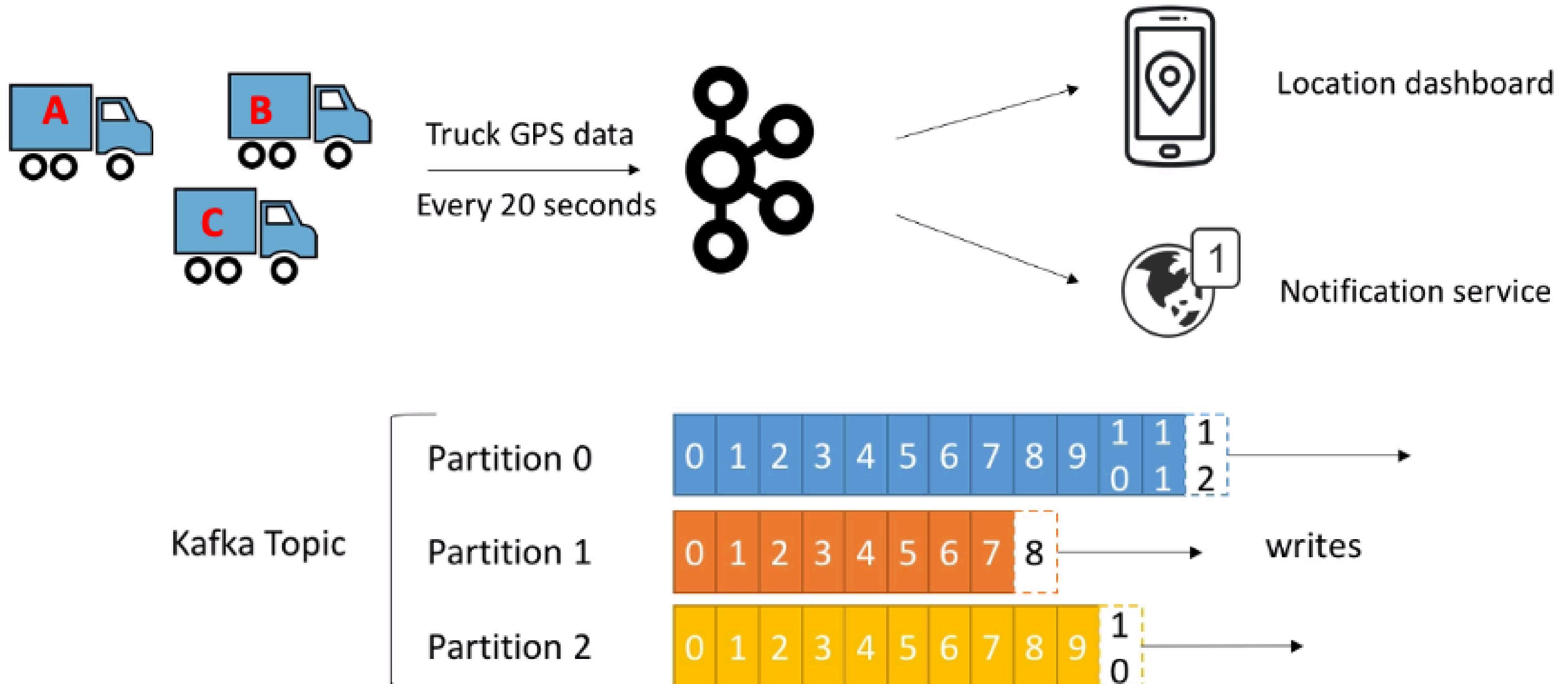


Topics y Particiones

- Cada partición es ordenada automáticamente
- Cuando se crea un topic por default
- Los tamaños de la partición pueden ser variantes
- la información por Kafka es retenida un tiempo temporal(defecto una semana) pero los offsets siguen aumentando
- La data en una partición son inmutables
- Cuando se crea un topic automáticamente asigna el topic y lo distribuye



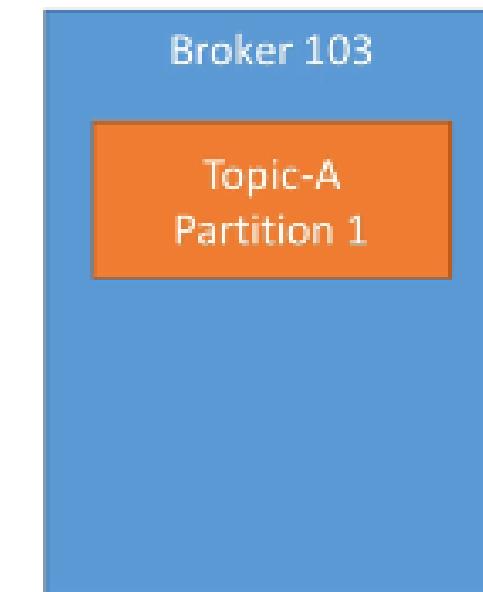
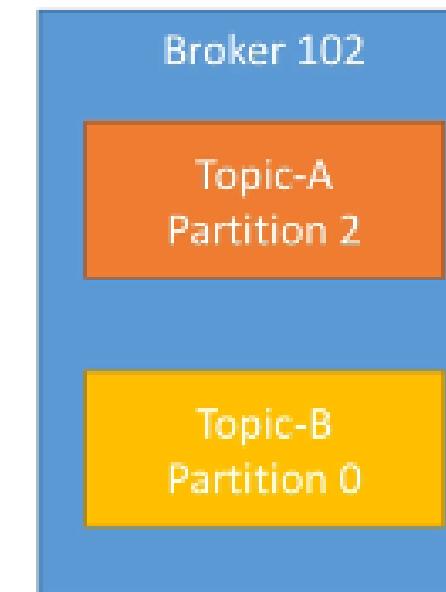
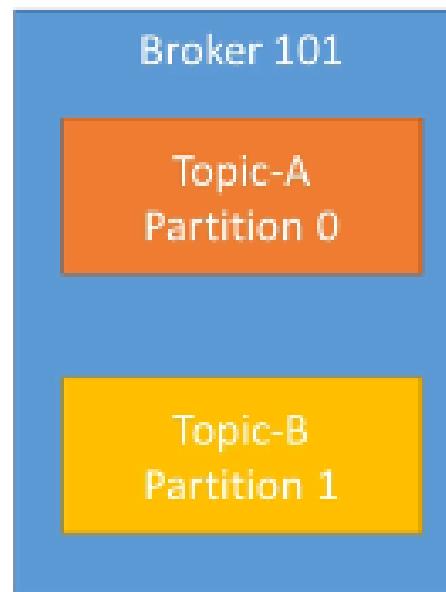
Ejemplo Kafka Con/Sin Partición



Brokers/Clusters

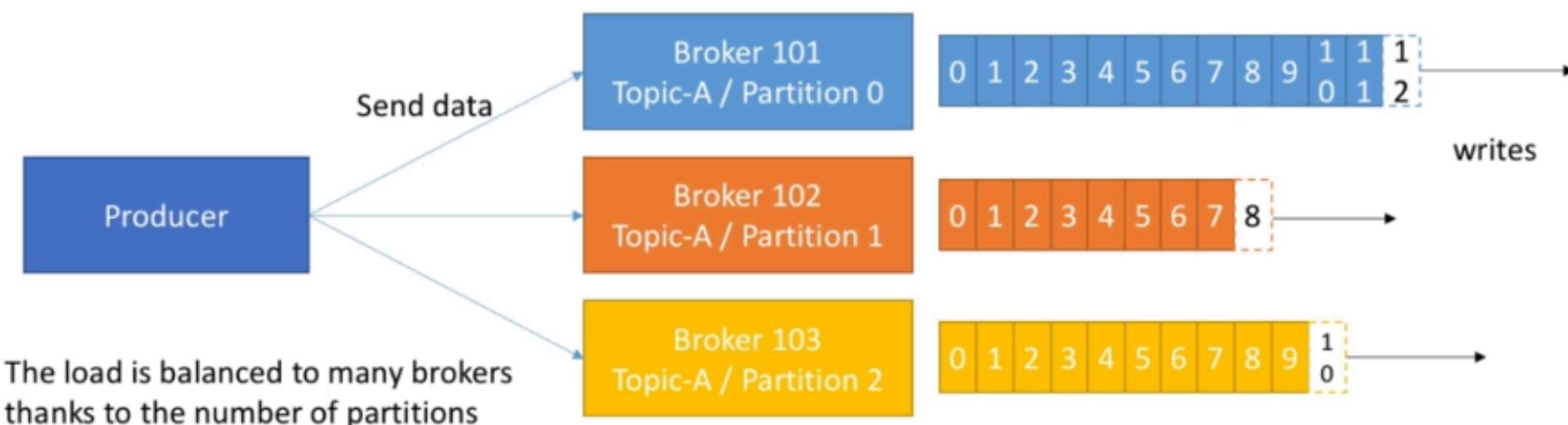
- Cada broker esta identificado por una identificación
- Cada broker tendrá solo ciertas particiones del topic
- Un buen numero de brokers es una cantidad de 3 brokers
- Después de conectado a cualquier bróker(llamado Bootstrap broker), ya te puedes conectar a todo el clúster

- Example of **Topic-A** with **3 partitions**
- Example of **Topic-B** with **2 partitions**



Producers

- Los productores escribe datos a los topics.
- Los productores automáticamente saben a qué broker y partición deben enviar sus mensajes
- En caso de que un broker falla, los productores se recuperan automáticamente



Los productores pueden recibir acks(confirmaciones), el cual es un recibido, es un sinónimo de confirmación:



Acks(0): Envía los datos y no esperan confirmación, en este caso hay posible perdida de dato



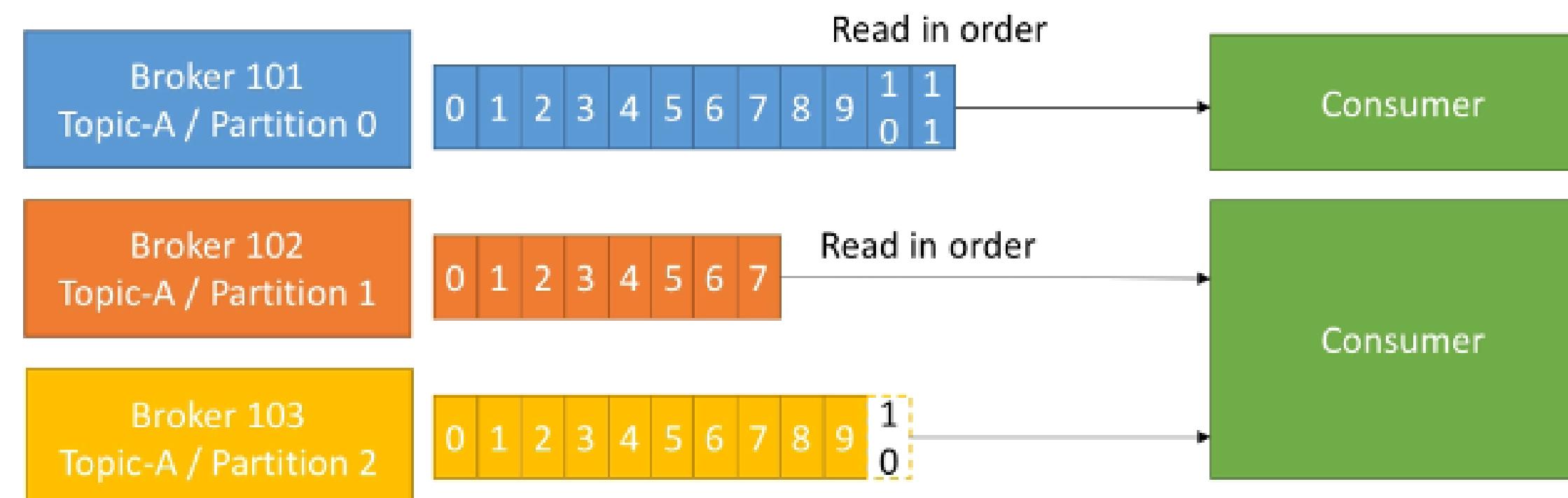
Acks(1): El productor espera que el bróker líder confirme, esto limita la perdida de información



Acks(all): EL líder y la replica deben decir que obtuvieron los datos, no genera perdida de datos

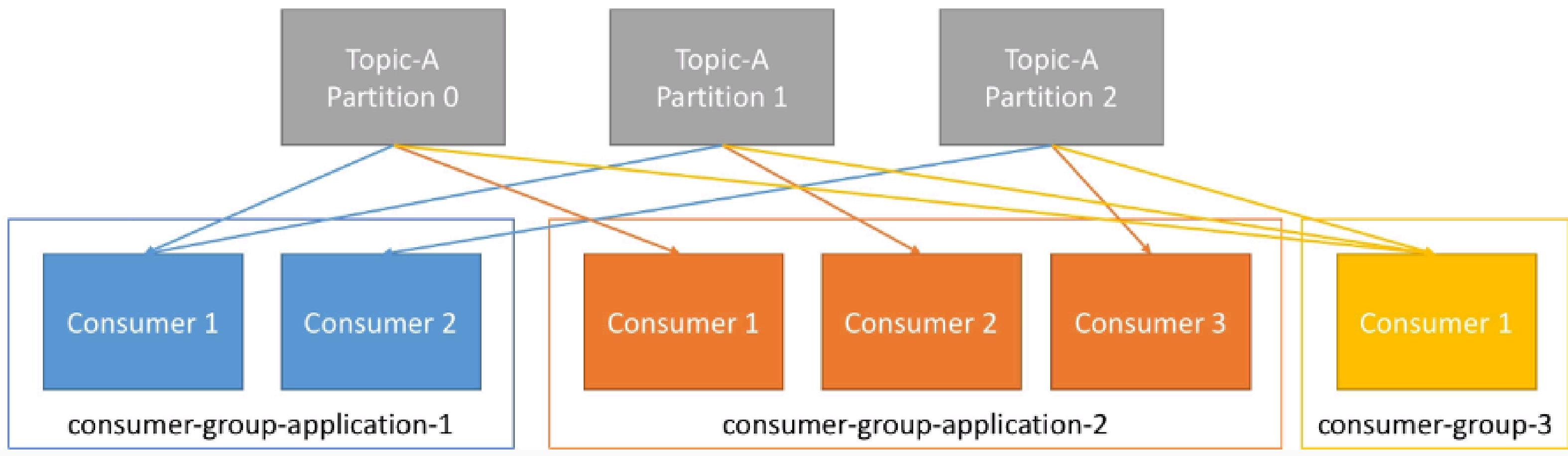
Consumers

- Los consumidores leen la información de un topic (identificado por un nombre)
- Los consumidores saben que broker leer automáticamente
- En caso de fallas los consumers sabrán como recuperarse
- Los consumers leerán los datos en orden en cada partición
- El consumidor no vera el subset 2 sin antes haber visto el 1
- Los datos se leerán en el orden dentro de cada partición
- Los consumidores también pueden leer en varias particiones, pero no hay garantía en el orden de la partición, los lee en paralelo



Consumer Groups

- Son consumidores de los datos, un grupo de consumidores representa una aplicación
- Cada consumer dentro de un grupo, lee exclusivamente de una partición
- Si tienes más consumers que partitions, algunos consumidores pueden estar inactivos
- Si se presentan problemas en uno de los consumers, el que esta inactivo toma el control inmediatamente
- se deben tener tantos consumidores como particiones a lo sumo



Preguntas

Gracias por escuchar.