Imagina

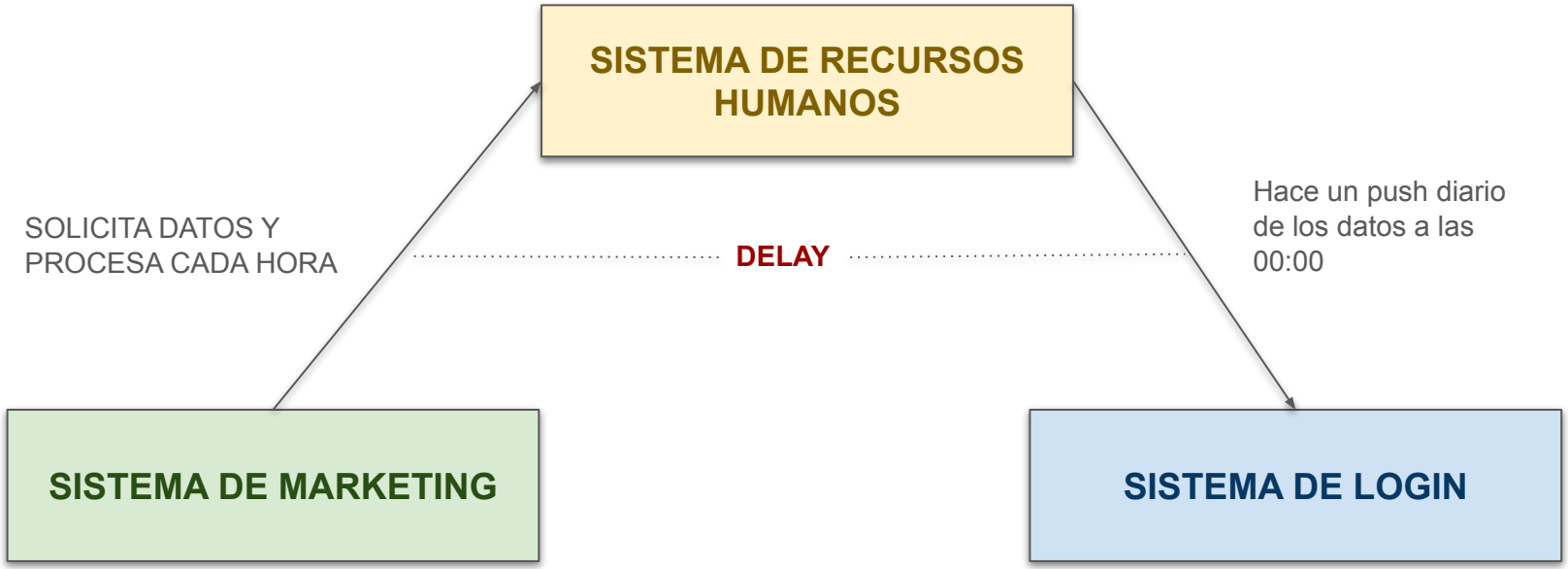
INTRODUCCIÓN A APACHE KAFKA PARA DESARROLLADORES JAVA

David Martínez Sepúlveda

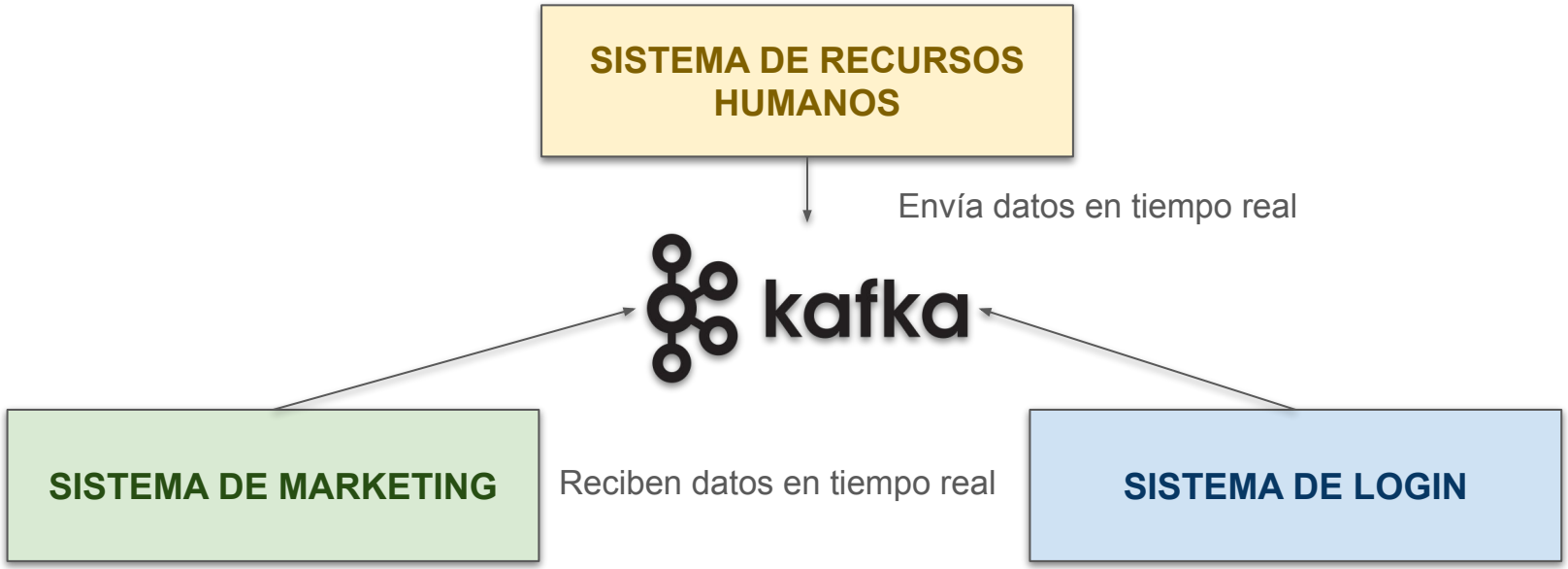
Sistema de Mensajería

- Comunicación entre aplicaciones.
- Gestiona mensajes dentro de una organización.
- Integración.

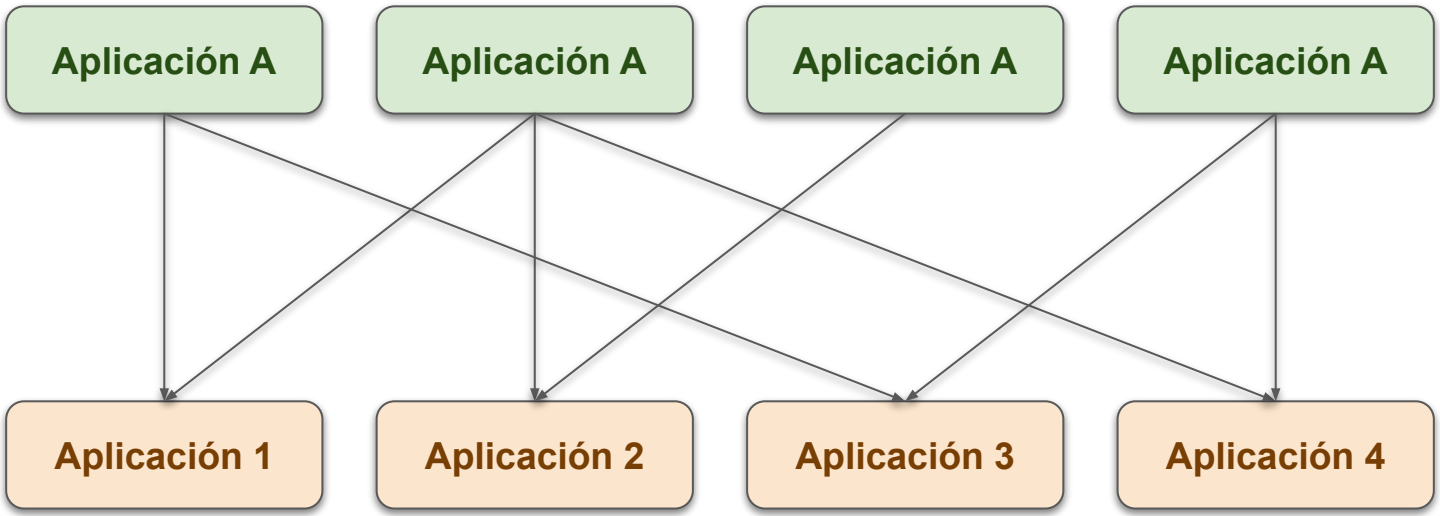
Sin Sistema de Mensajería



Con Sistema de Mensajería



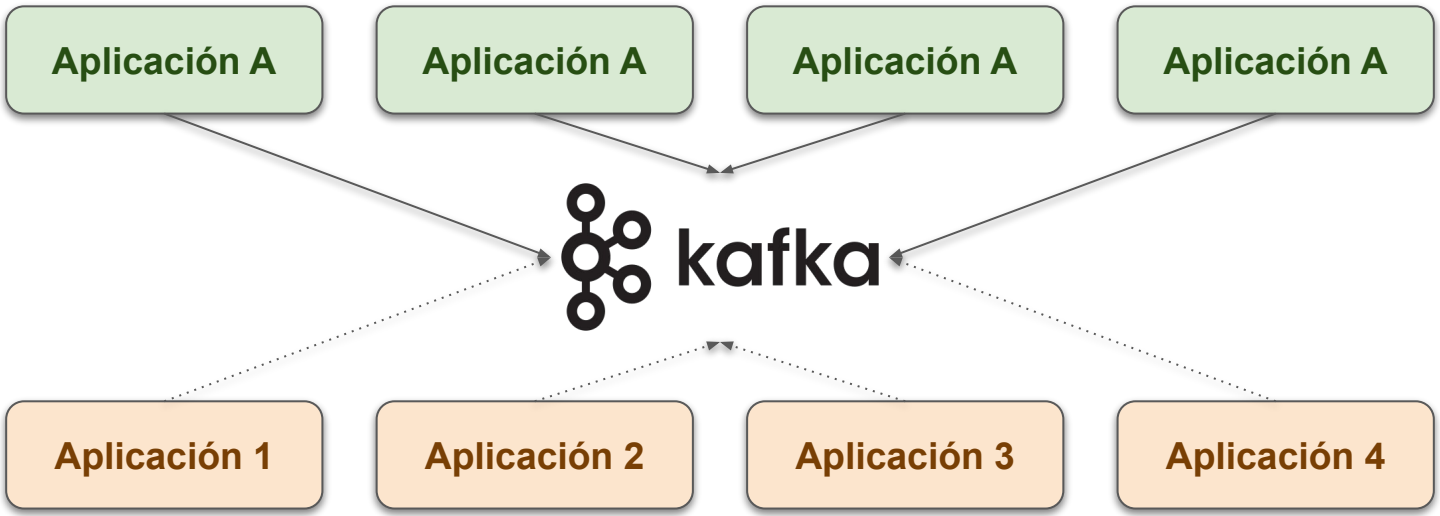
Sin Sistema de Mensajería



Problemas de esta implementación

- Diversos puntos de integración.
- Diferentes implementaciones.
  - Protocolo de datos: enlace a BD, API Rest, ficheros…
  - Formato de datos: binario? JSON? CSV? XML?
- Mantener conexiones.

Kafka como Sistema de Mensajería



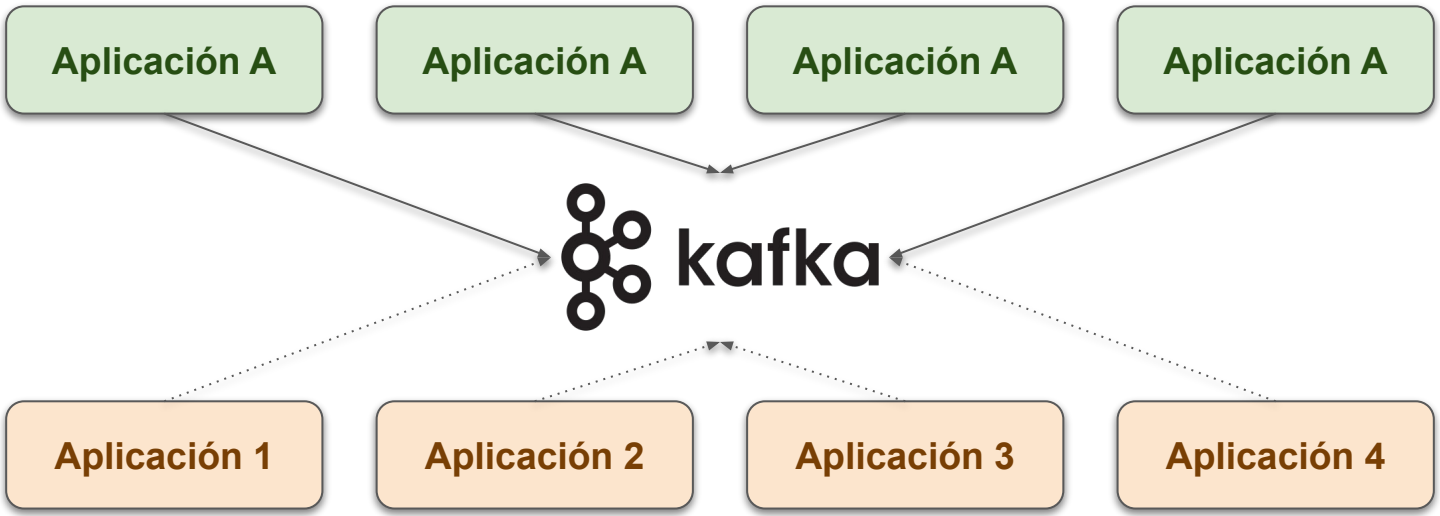
Casos de uso

- Por cada pago recibido: crear un historial de pagos, enviar notificación al usuario, informar a logística para enviar el pedido.
- Por cada detección de posible fraude: suspender temporalmente la cuenta, enviar un email al auditor, congelar la transacción.
- Evitar el bloqueo de recursos en procesos que requieran mucho tiempo.

¿Qué es Apache Kafka?

- Kafka es una plataforma de transmisión de eventos.
- Características:
  - **Publicar y consumir eventos**, incluyendo importación y exportación continua de datos desde otro sistema.
  - **Almacenar transmisiones de eventos** de manera permanente y confiable el tiempo que deseemos.
  - **Procesar transmisiones de datos** cuando ocurren o de manera retrospectiva.

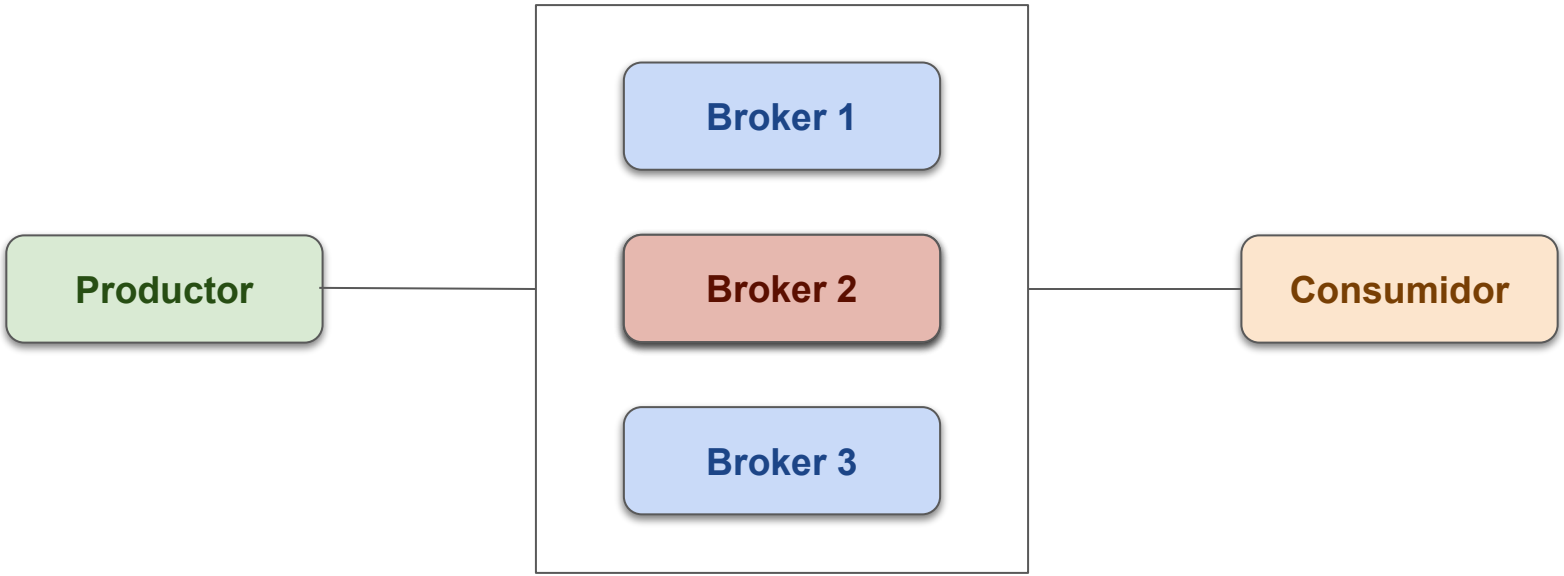
Publicar y consumir eventos



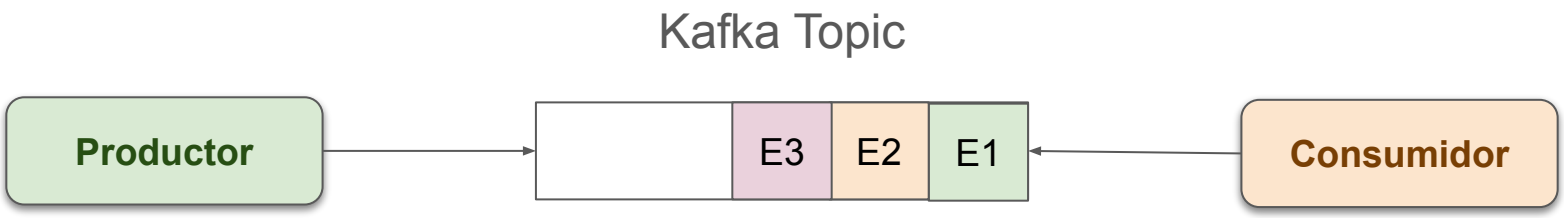
Clúster de Kafka

- Una máquina que ejecuta Kafka se conoce también como un servidor Kafka o **Broker**.
- Se puede usar un único bróker.
- Un **cluster** consiste en *varios brokers tratados como una unidad* para alto rendimiento y disponibilidad.

Publicar y consumir eventos



Publicar y consumir eventos



Arquitectura básica de Kafka

- **Producers**: Publican datos en topics.
- **Topics**: Canales donde los datos se agrupan en particiones para escalabilidad.
- **Consumers**: Suscriben y leen datos desde los topics.
- **Brokers**: Servidores que gestionan los datos dentro del clúster.

Casos de uso de Kafka

- Sistema central de mensajería.
- Log de actividades/aplicaciones.
- Almacenar datos de IoT.
- Desacoplamiento de sistemas.
- Procesado asíncrono.
- Parte de un ecosistema big data.

Logs distribuidos y durabilidad de los mensajes

- **Logs distribuidos**: Kafka almacena mensajes como un registro inmutable y ordenado en disco.
- **Durabilidad**:
  - Replicación.
  - ACKs configurables.
  - Retención configurable.

Integración de Kafka con Spring Boot

Spring Boot facilita la integración con Kafka mediante el módulo **Spring Kafka**, que proporciona:

- Configuración automática para productores y consumidores.
- Anotaciones declarativas como *@KafkaListener* para definir consumidores.
- Soporte para procesamiento de errores y reintentos automáticos.

API de productor y consumidor de Kafka en Java

- **Producer API**: Permite enviar datos a Kafka de manera eficiente, definiendo particiones, claves y valores.
- **Consumer API**: Gestiona la suscripción a topics y lectura de mensajes.

Configuración de Kafka en un entorno de desarrollo

- Instalación manual.
- **Uso de Docker y Docker Compose**.
- Servicios gestionados.
- Entornos de desarrollo preconfigurados.

Kafka vs RabbitMQ

Aspecto	Kafka	RabbitMQ
Modelo	Logs distribuidos	Colas tradicionales
Persistencia	Almacenamiento por defecto	Opcional
Escalabilidad	Horizontal por diseño	Limitada por arquitectura
Casos de uso	Streaming de datos	Mensajería tradicional

Instalación y configuración de un clúster Kafka

Objetivos del clúster de pruebas

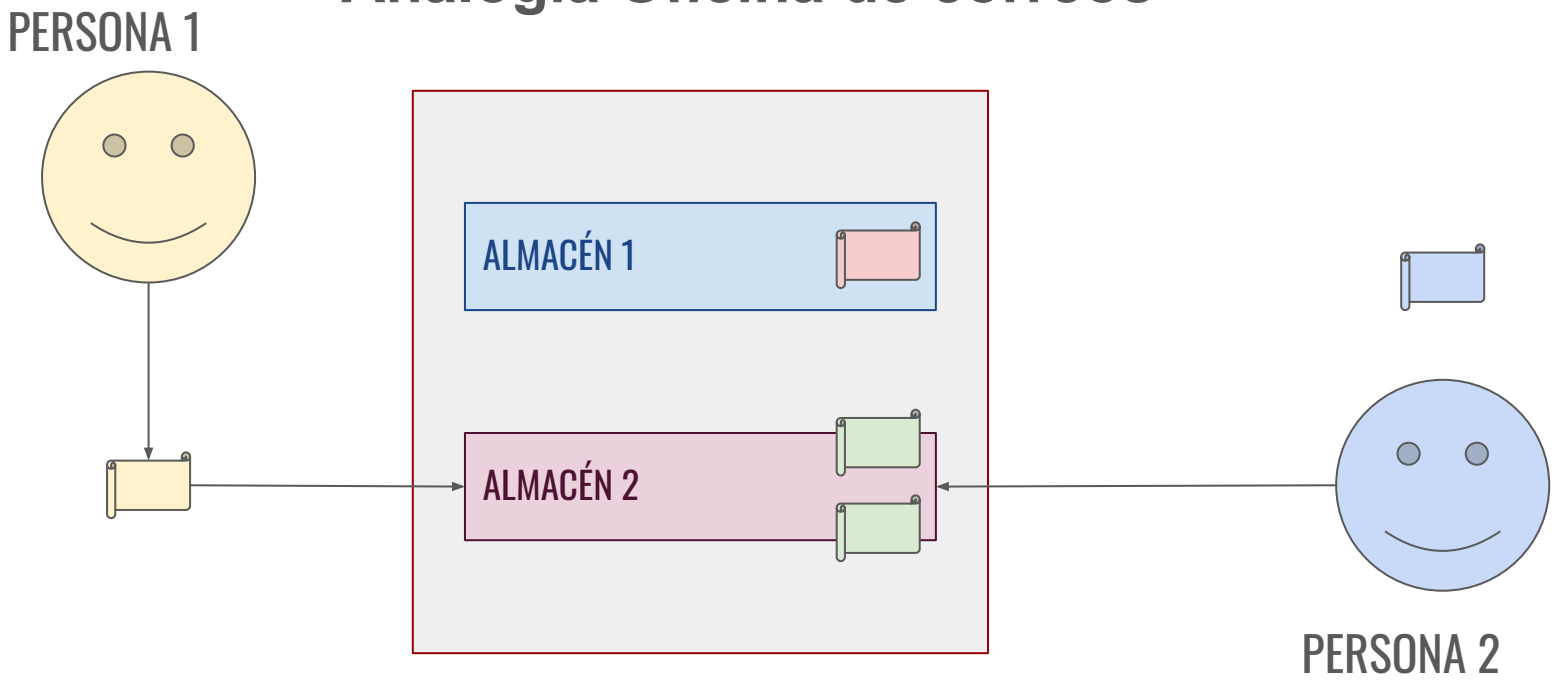
- Simular entornos reales.
- Desarrollar y probar aplicaciones.
- Facilidad de configuración y limpieza.

Instalación y configuración de un clúster Kafka

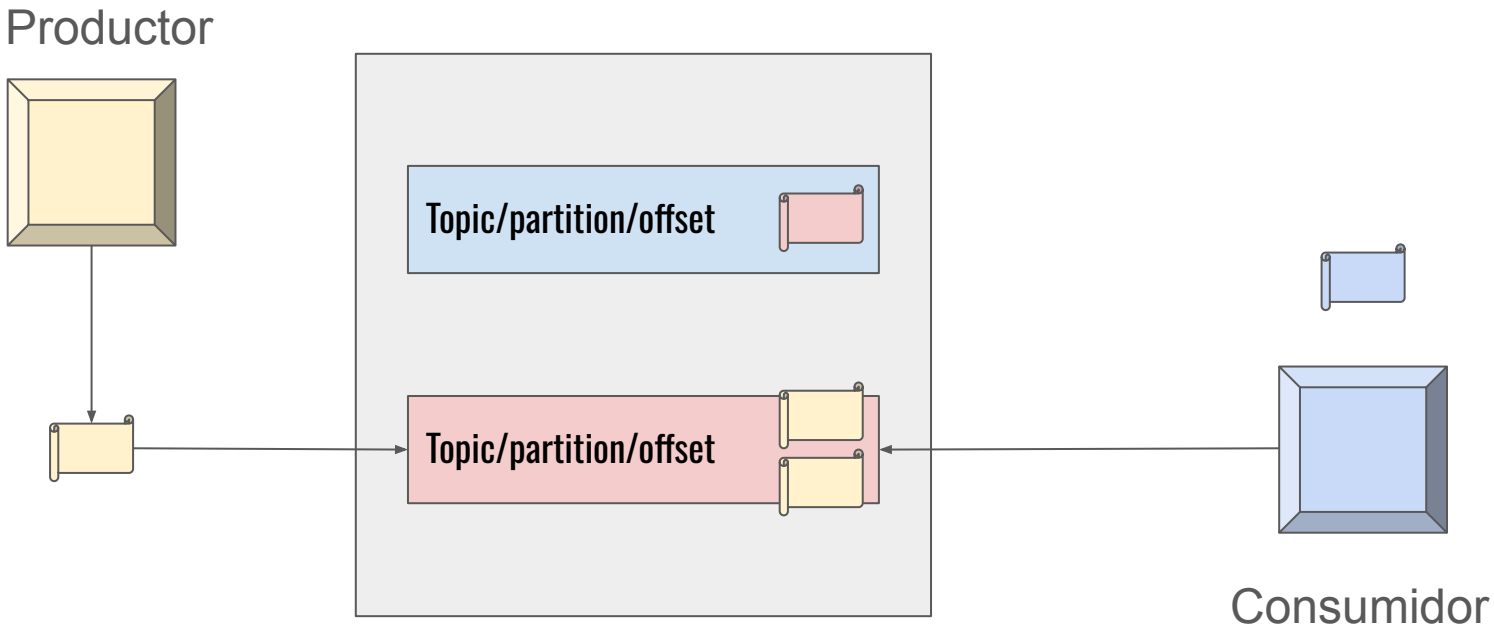
Ventajas de esta configuración

- Escalabilidad.
- Tolerancia a fallos.
- Flexibilidad.

Analogía Oficina de correos



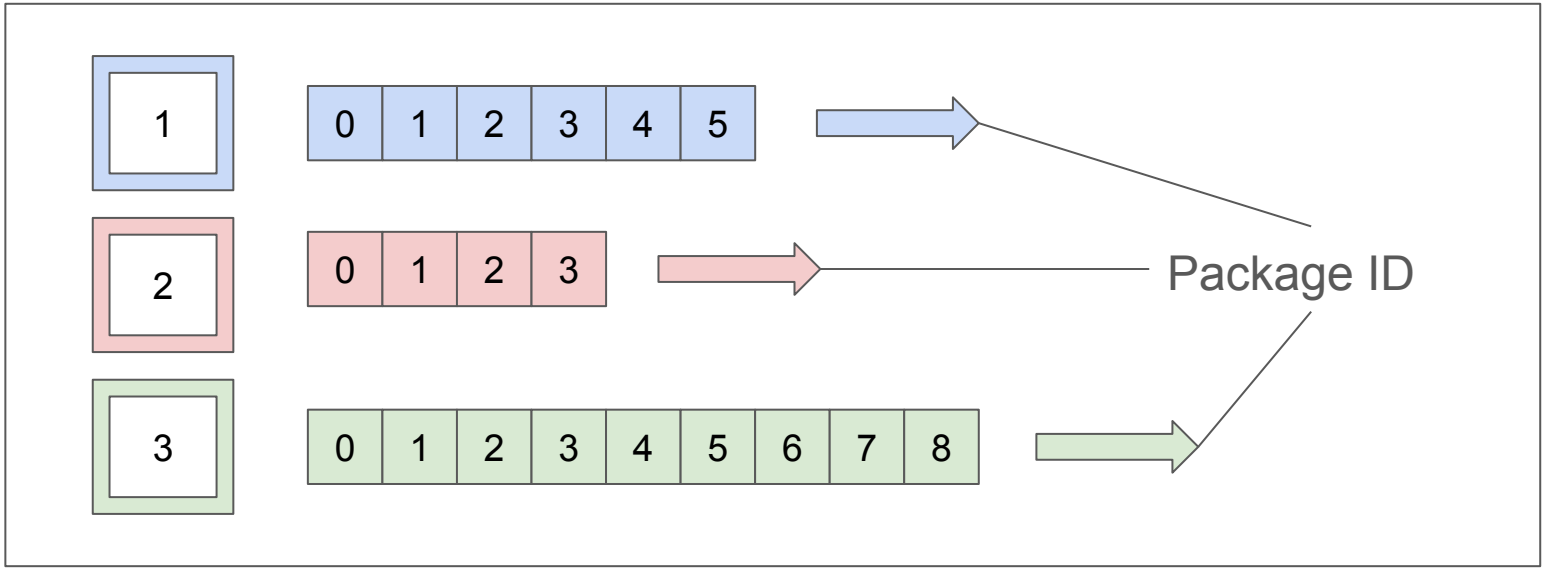
Analogía Oficina de correos



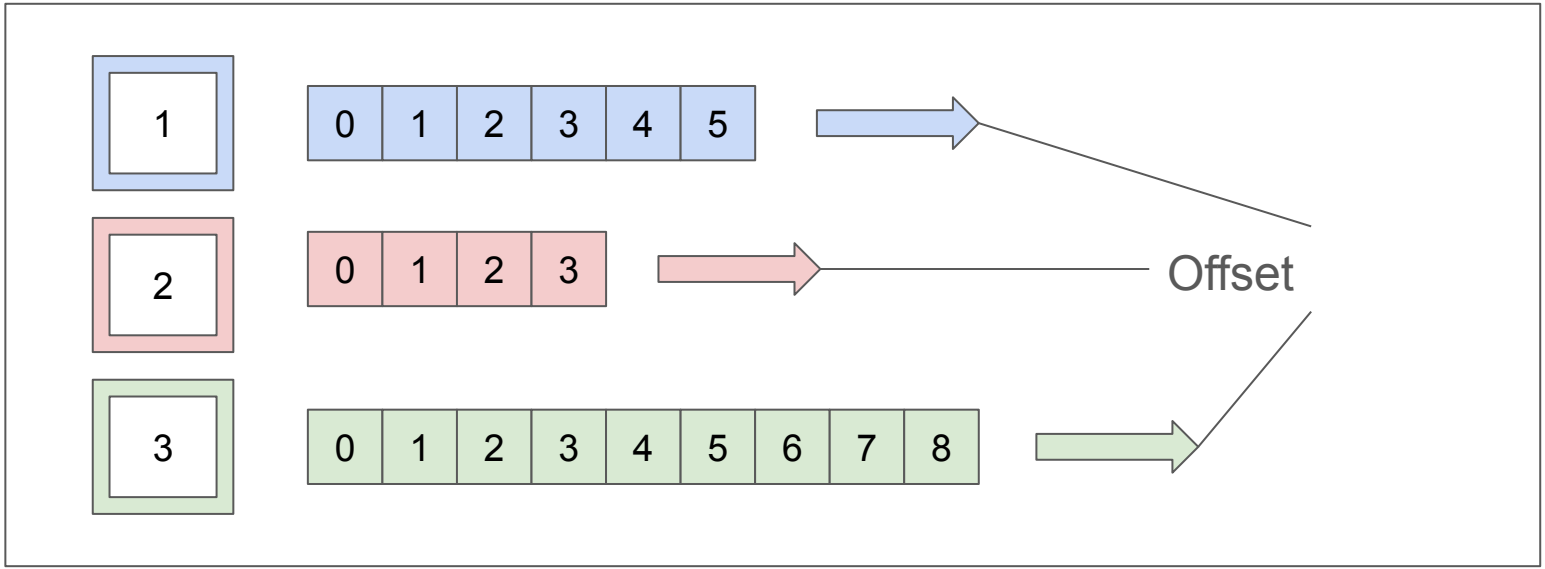
Terminología

- Producer/publisher
- Send/publish/produce message
- Subscriber/consumer/listener
- Subscribe/consume/listen

Analogía Oficina de correos



Analogía Oficina de correos



Topics, particiones y offset

- Podemos tener tantos topics como necesitemos
- Los mensajes se almacenan por un determinado período
  - 7 días por defecto
  - Se puede configurar
- Cada topic tiene un nombre
- Los mensajes son inmutables

Topics, particiones y offset

- 1 topic: 1 o más particiones
- Las particiones proporcionan paralelismo
- Los mensajes se almacenan en orden para cada partición
- El orden no está garantizado entre particiones

Topics, particiones y offset

- Cada mensaje tienen un offset.
- El offset empieza en 0
- Offset por partición
  - Partición 0, offset 0|1|2|3
  - Partición 1, offset 0|1|2|3
- Cada partición es independiente

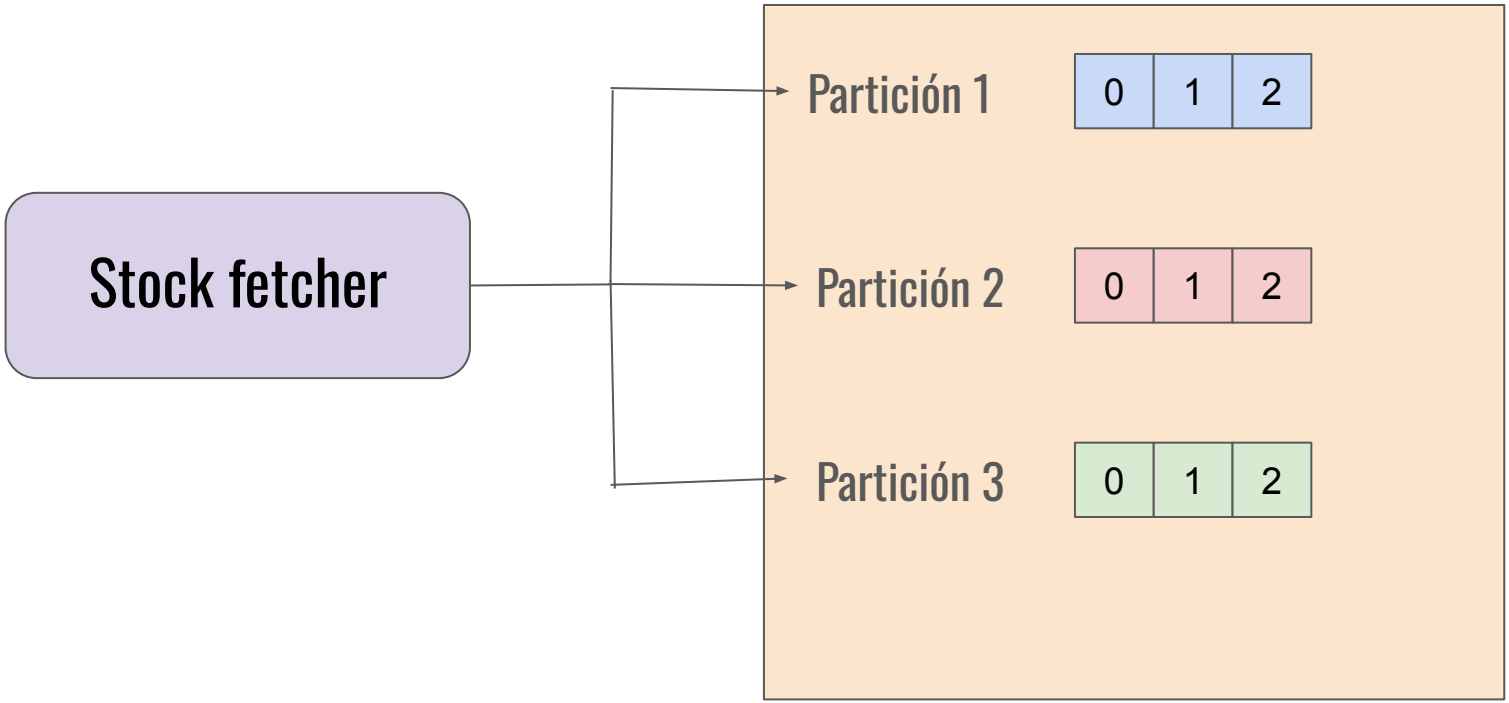
Topics, particiones y offset

- Las particiones se definen al crear el topic.
- Se pueden agregar particiones una vez creado.
- No se pueden borrar particiones una vez creado.

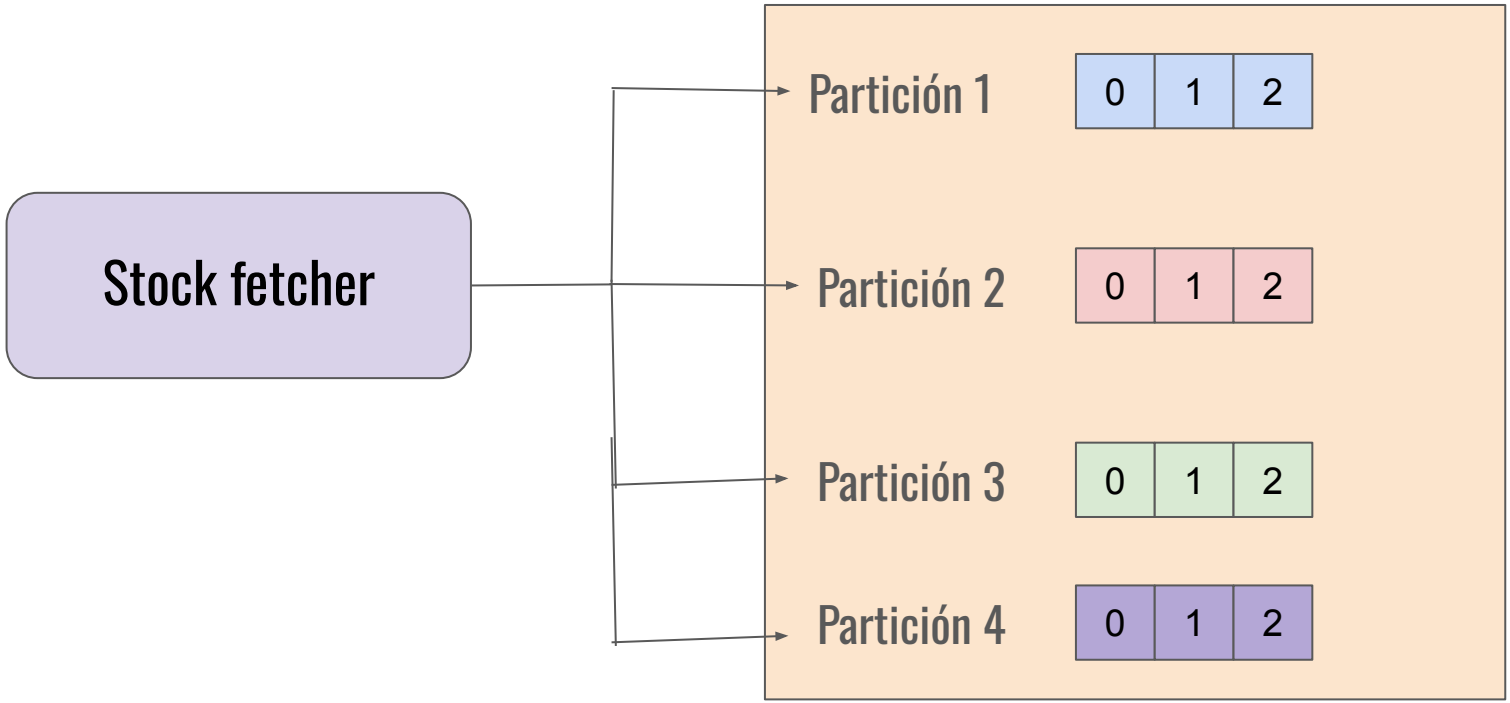
Productor

- Envía mensajes a Kafka.
- Elige el topic y el contenido del mensaje.
- Kafka selecciona la partición automáticamente pero se puede sobreescribir este comportamiento.

Productor


Stock fetcher
Partición 1
0
1
2
Partición 2
0
1
2
Partición 3
0
1
2

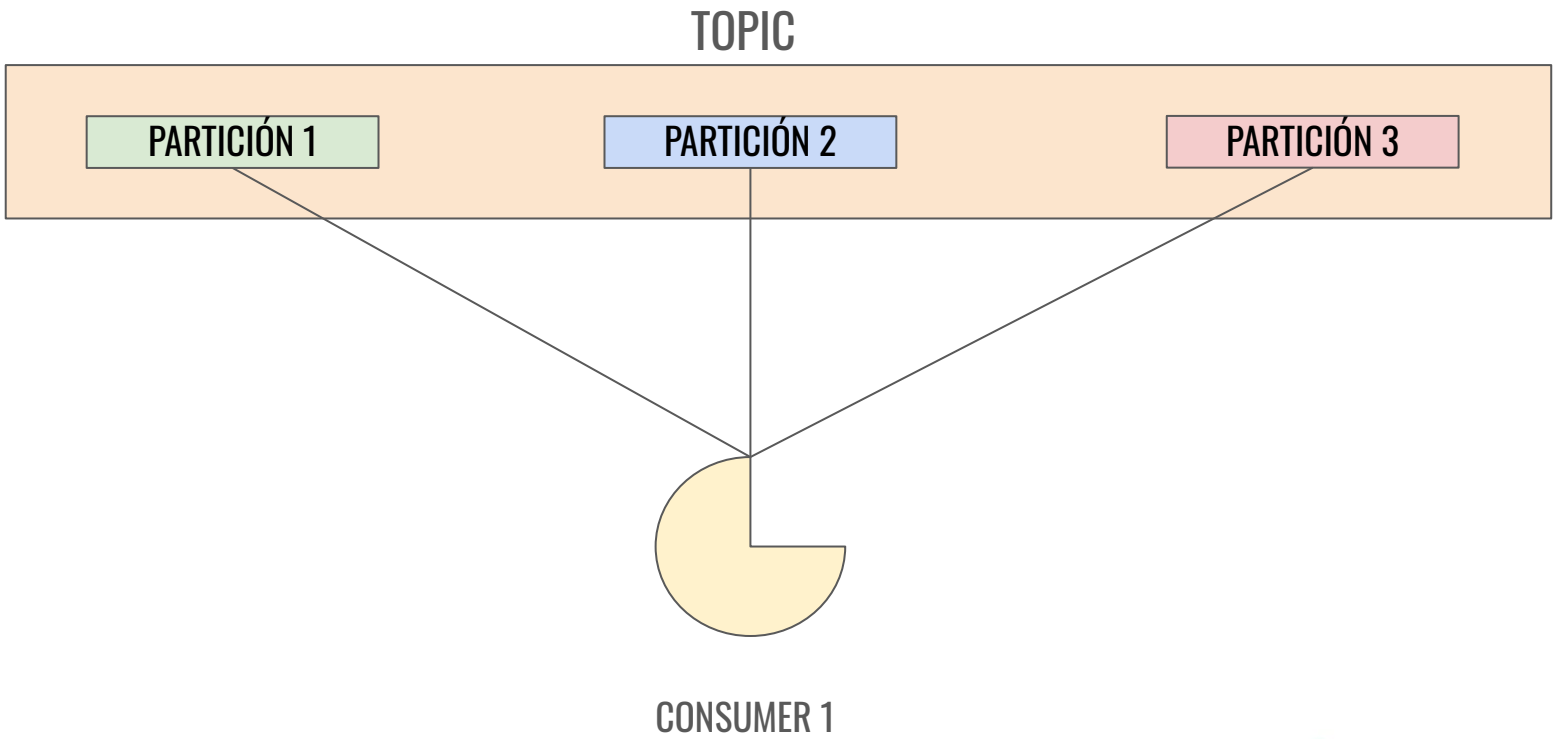
Productor


Stock fetcher
Partición 1
0
1
2
Partición 2
0
1
2
Partición 3
0
1
2
Partición 4
0
1
2

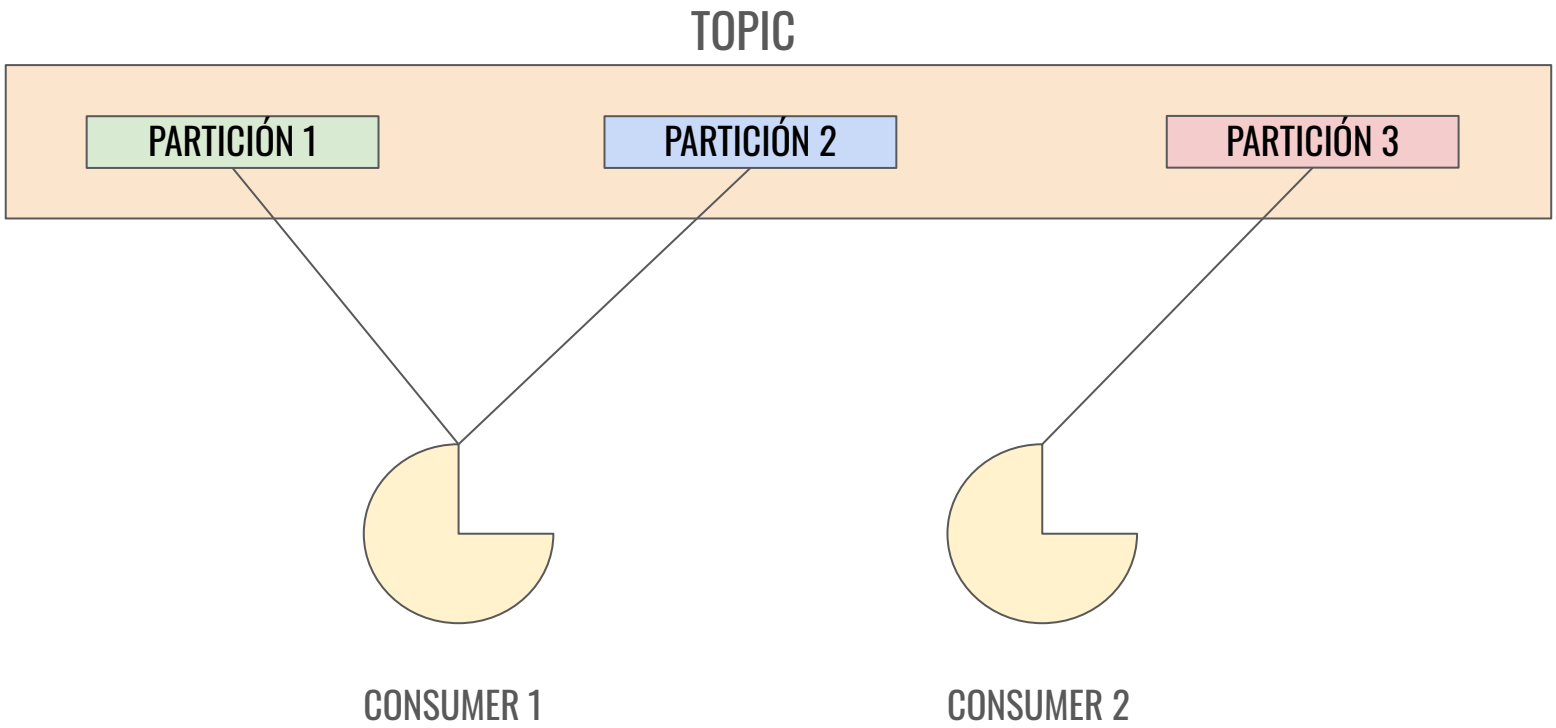
Consumer

- El consumer lee los datos en orden para cada partición.
- El orden se determina por el offset (de menor a mayor).
- Cada partición tiene un máximo de un consumer por consumer group.
- Un consumer puede leer de más de una partición.

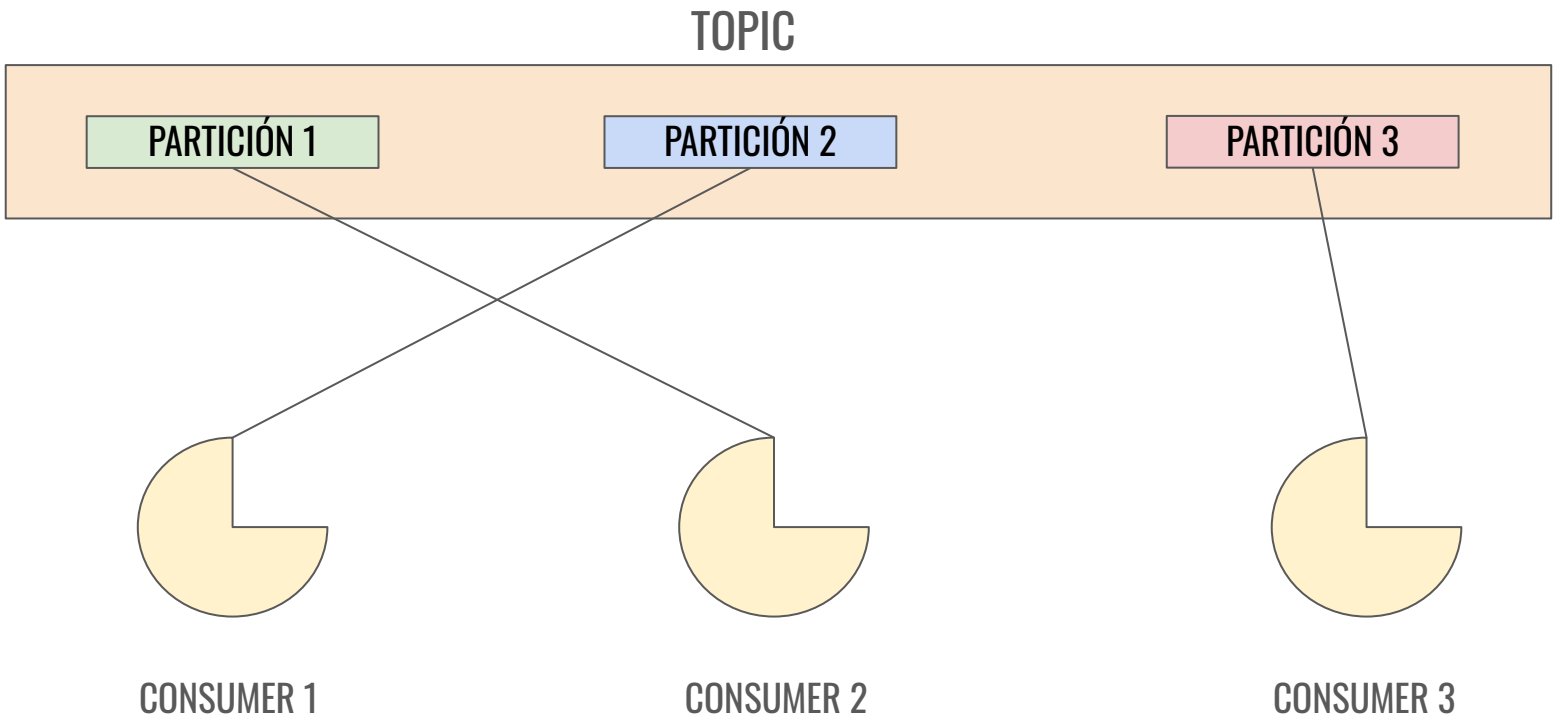
Consumer



Consumer


TOPIC
PARTICIÓN 1
PARTICIÓN 2
PARTICIÓN 3
CONSUMER 1
CONSUMER 2

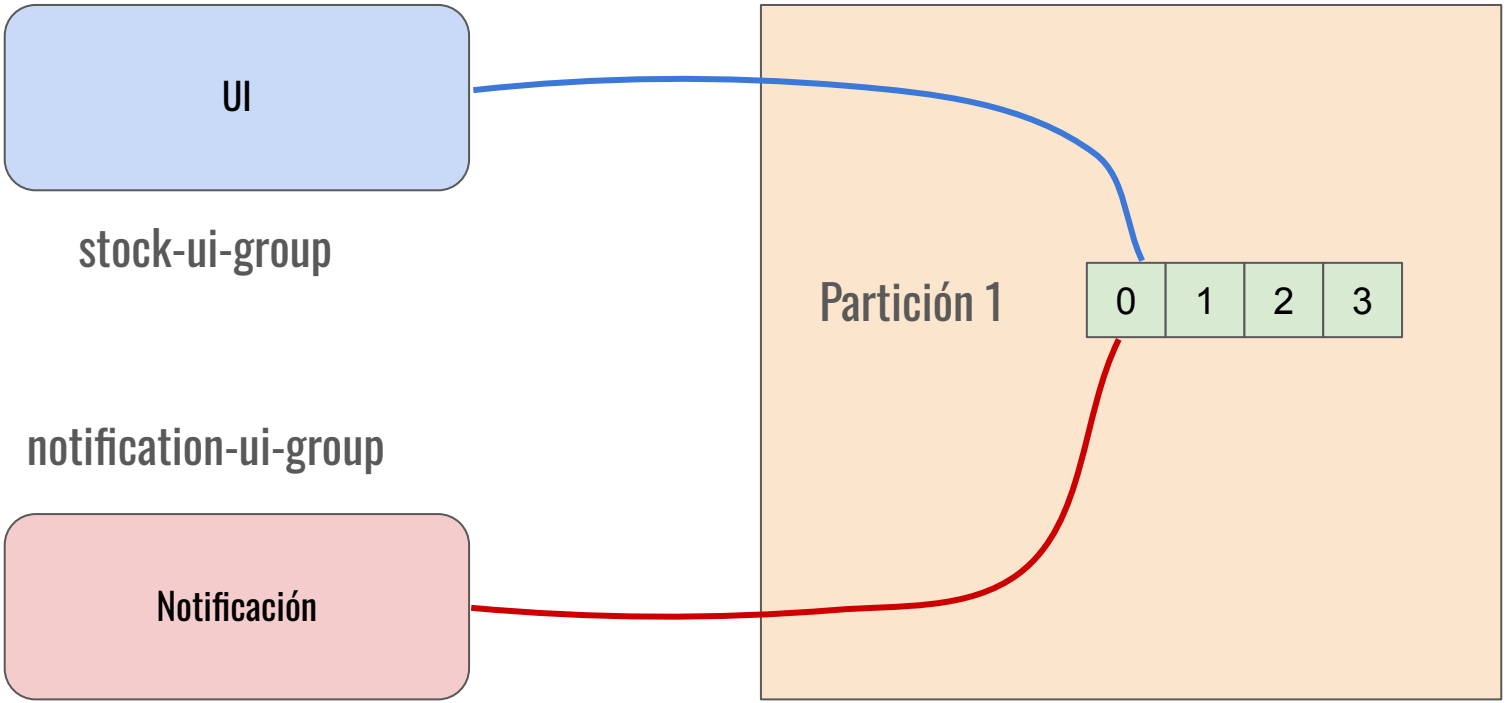
Consumer


TOPIC
PARTICIÓN 1
PARTICIÓN 2
PARTICIÓN 3
CONSUMER 1
CONSUMER 2
CONSUMER 3

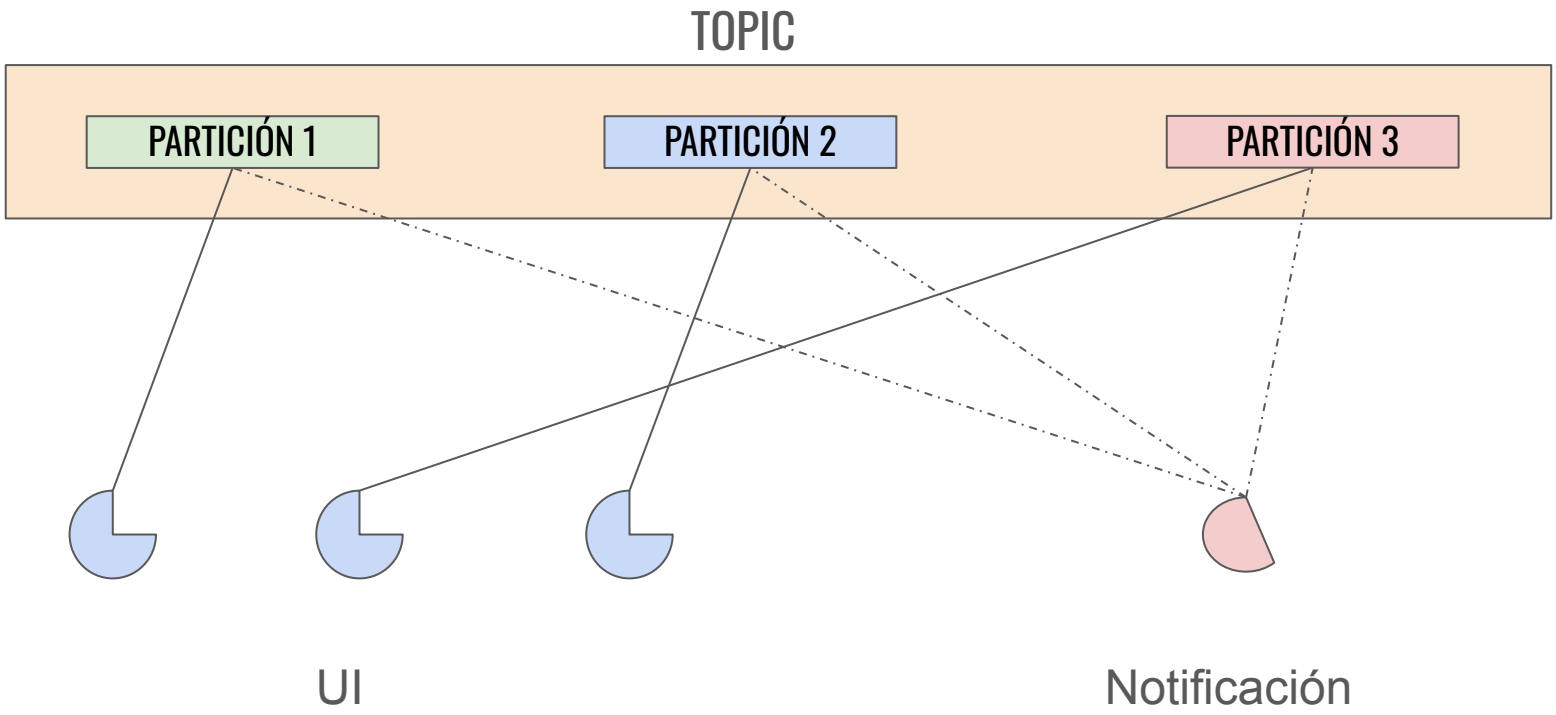
Consumer Group

- Agrupa a los consumidores basándose en funcionalidades.
- Consumo en paralelo.

Consumer group



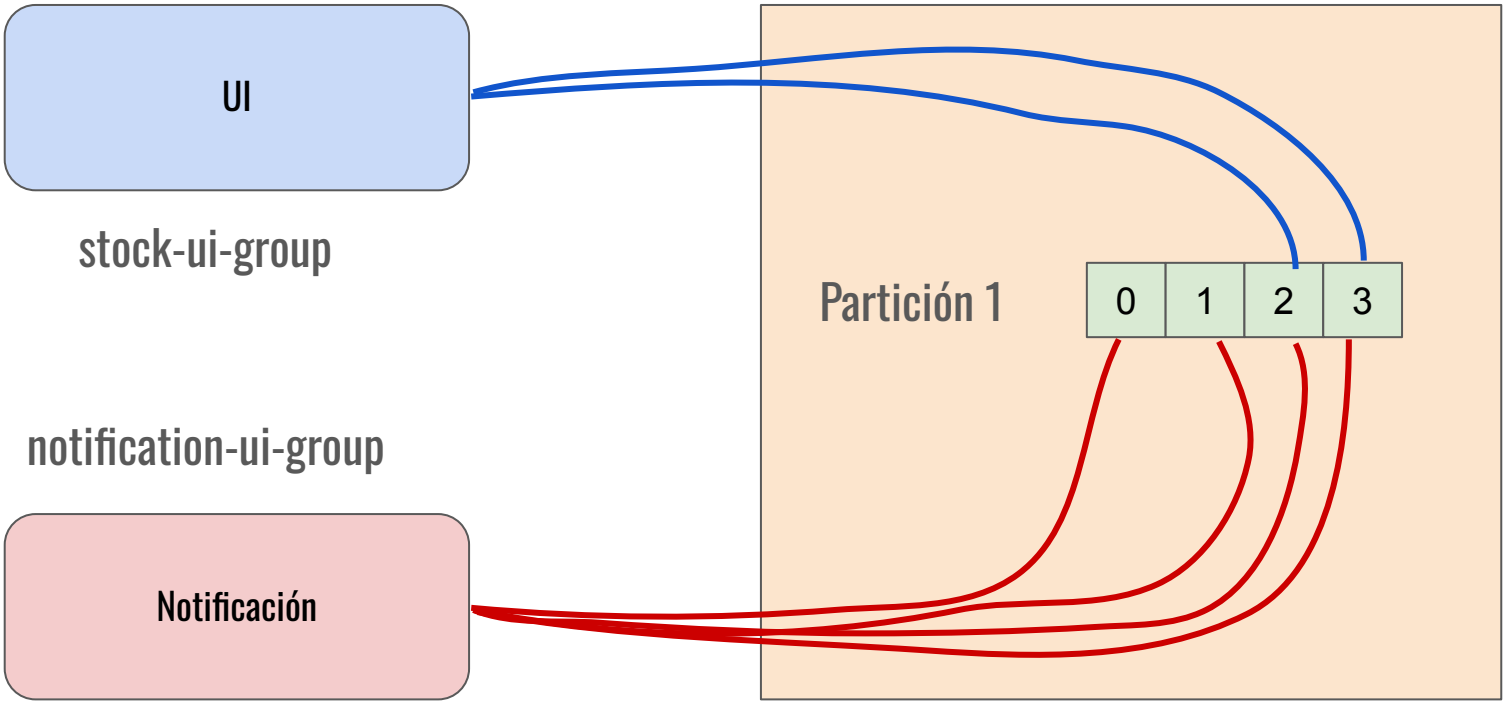
Consumer group



Consumer Offset

- Almacena el offset que ha sido leído.
- Es único para cada partición.
- Kafka almacena el offset del consumer.
- Es un punto de control que indica el último punto leído.

Consumer Offset



Consumer Offset

- Los consumidores eligen cuando guardar el offset.
- At-most-once.
  - 0 (sin procesar) o 1 (procesado)
  - Se pueden perder mensajes
- At-least-once
  - Una o más
  - Si algo falla, se puede reprocesar
  - Crear un consumidor idempotente.

Consumer Offset

- Exactly-once:
  - Sólo una vez
  - Difícil de implementar

Ejercicio práctico: Crear una aplicación Java que envíe y consuma mensajes en Kafka

- Crea un topic llamado `t-numbers`
- Crea un productor que produzca números aleatorios cada segundo entre 1 y 15.000
  - `int randomNumber = (int) (Math.random() * 15000) + 1;`
  - `TimeUnit.SECONDS.sleep(1);`
- Crea un consumidor que consuma los números e indique si el número es par o impar.


Imagina