# Kafka 101

Marco Robles Pulido

@marcoantoniorob

# Important Notes

Identify yourself in Zoom, using your name and last name

Mute your microphone along the course

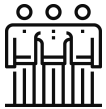
Use the chat for questions during the Q&A sections

Focus your questions on the presented topic

Turn off your camera in case of connection issues

# Academy Code of Conduct

Be respectful, there are no bad questions or ideas.



Be welcoming and patient

Be careful in the words that you choose

# Objetivos de la sesión

**Al final de la sesión seremos capaces de:**

- Entender el propósito de Kafka
- Qué es un sistema pub/sub
- Entender los conceptos base de Kafka (consumidores, productores y tópicos)
- Saber cuándo podemos usar Kafka

# Table of Contents


**Recap**
Resumen de conceptos base de Kafka

**Topics & Brokers**
Manejo de un tópico dentro de Kafka

**Producers**
Función de un productor en Kafka

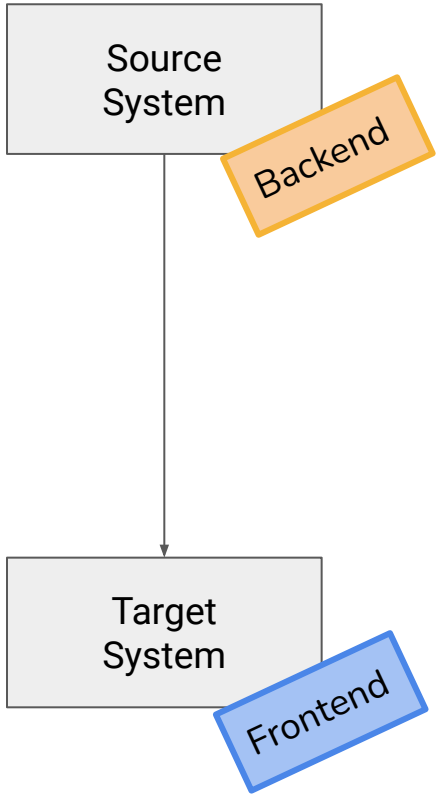
**Consumers**
Función de un consumidor en Kafka

**Consumer Group & Offsets**
¿Qué pasa cuando agrupo a mis consumidores?

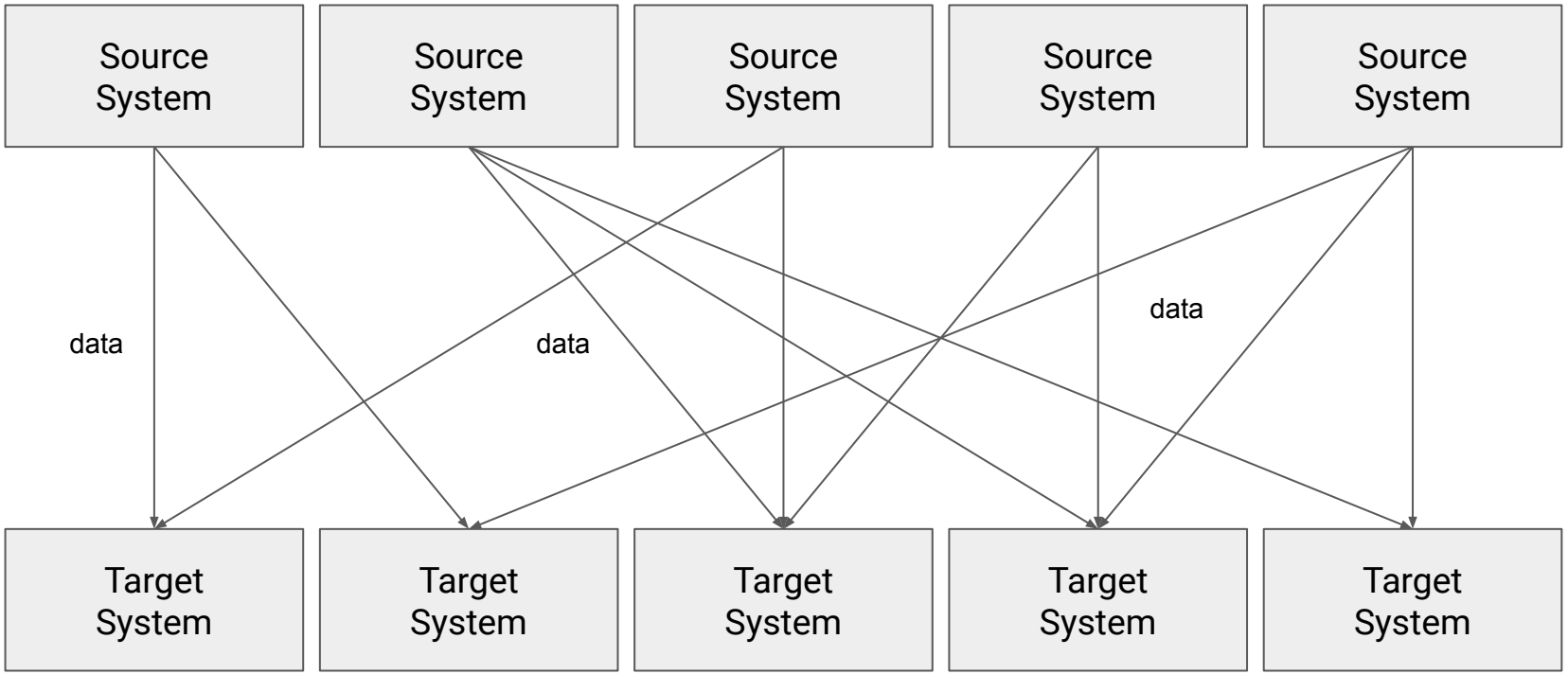
# Recap

Conceptos base de Kafka

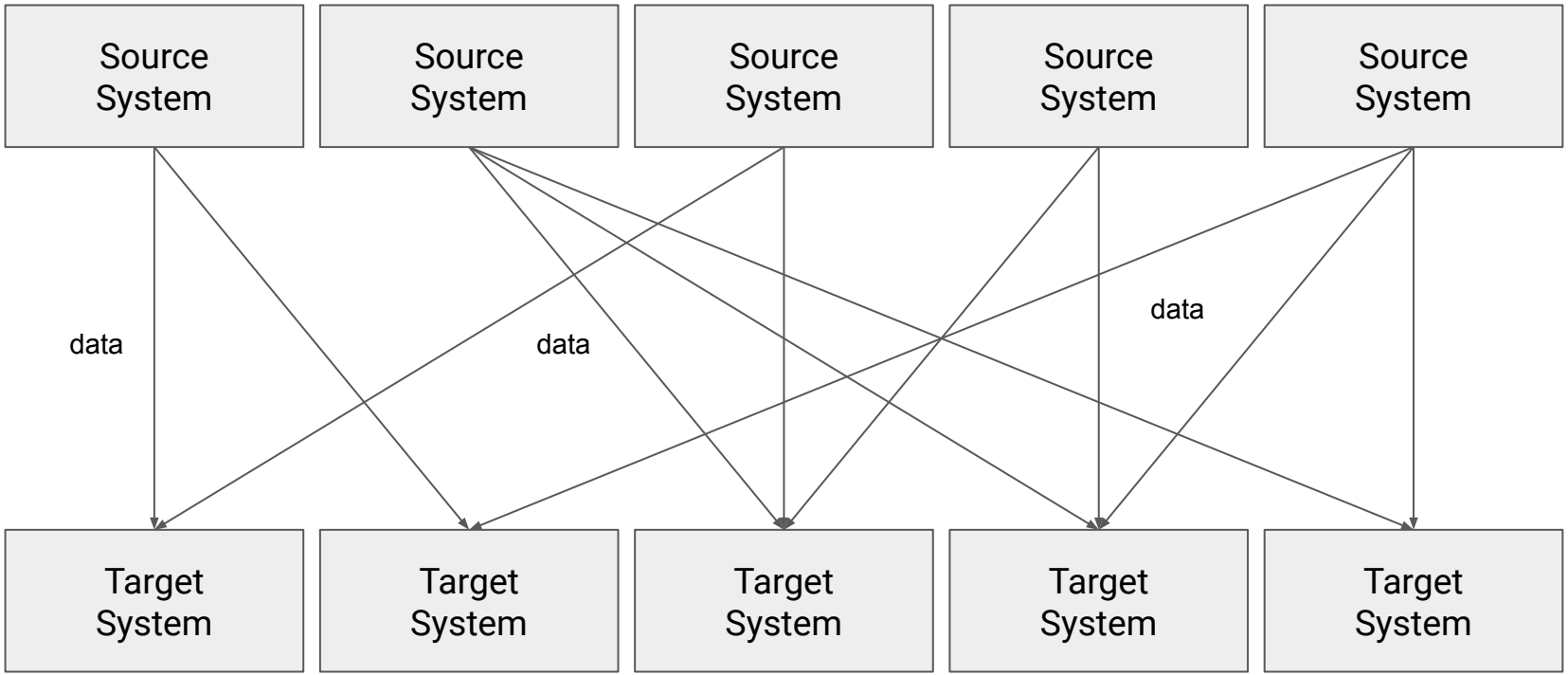
# Cómo empieza una organización...



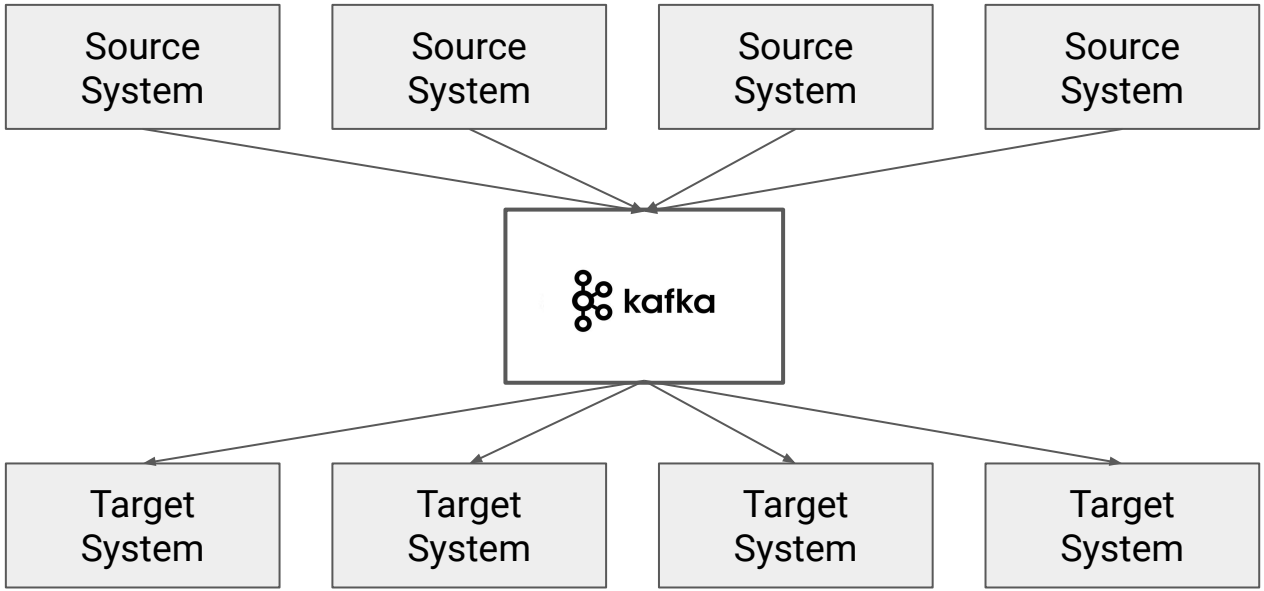
# Cómo termina una organización...



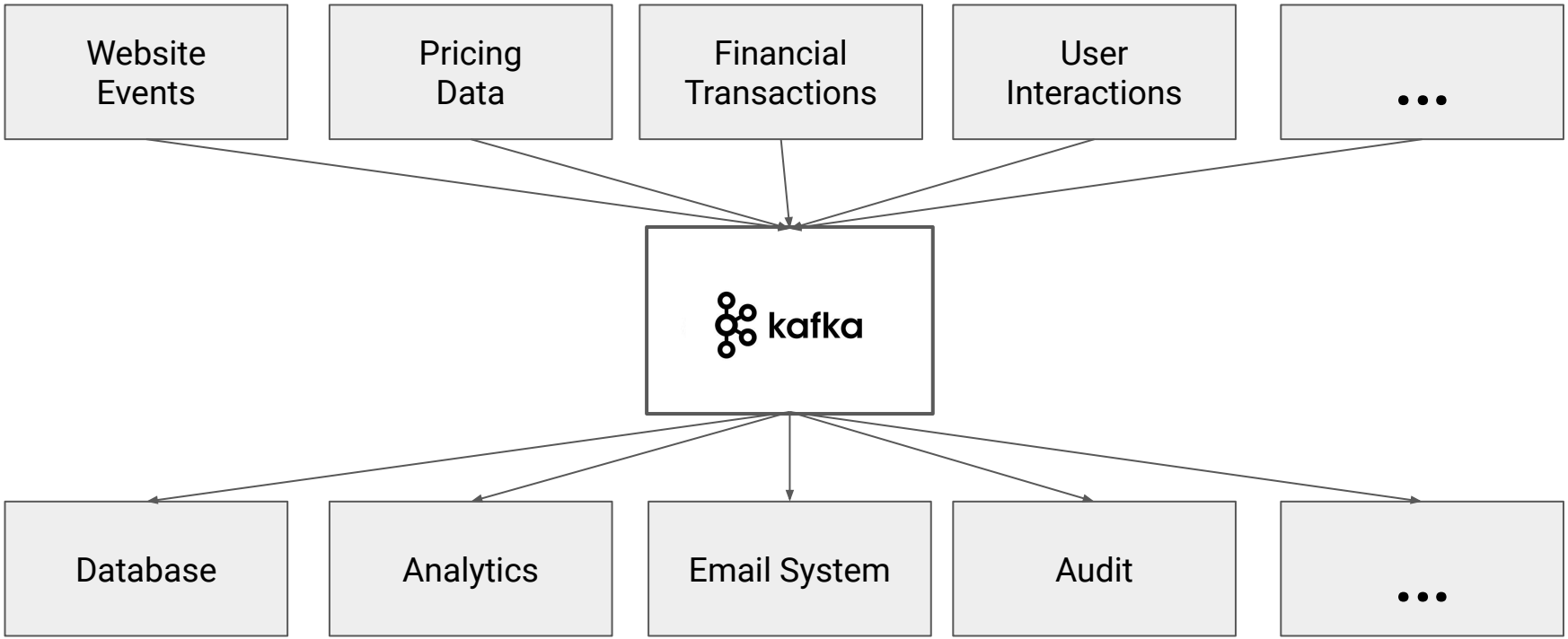
# ¿Problema?



# ¡Kafka al rescate!




# ¡Un ejemplo!



# ¿Por qué Kafka?

- Creado por LinkedIn y es un **proyecto open-source**
- Mantenido por **Confluent** y administrado por **Apache org**
- **Distribuido**
- **Arquitectura resiliente**
- **Tolerante a fallos**
- **Escalabilidad horizontal** (ej. Clusters de 100 brokers)
- Usado por **diversas empresas** como:

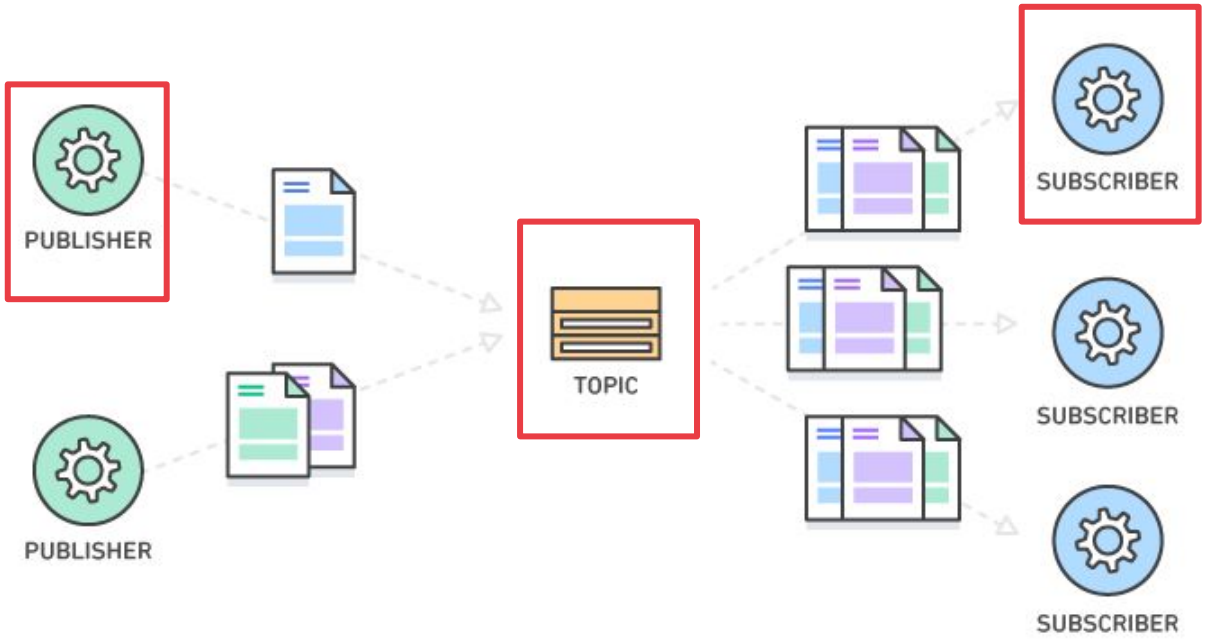
## Casos de Uso

- Sistema de mensajería
- Seguimiento de actividad
- Recopilación de métricas
- Recopilación de logs de la aplicación
- Procesamiento en streaming
- Desacoplamiento de sistemas
- Integración con otros sistemas para Big Data

# Pub/Sub System

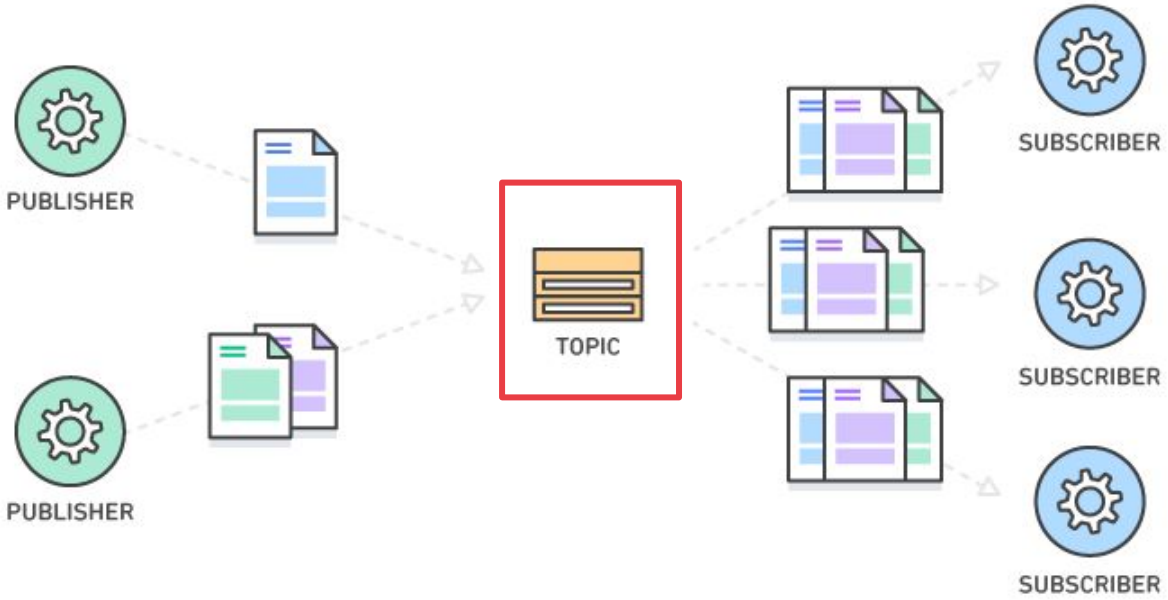
# Publisher/Subscriber

Sistema de mensajería (**pub/sub**) con enfoque al uso de colas.

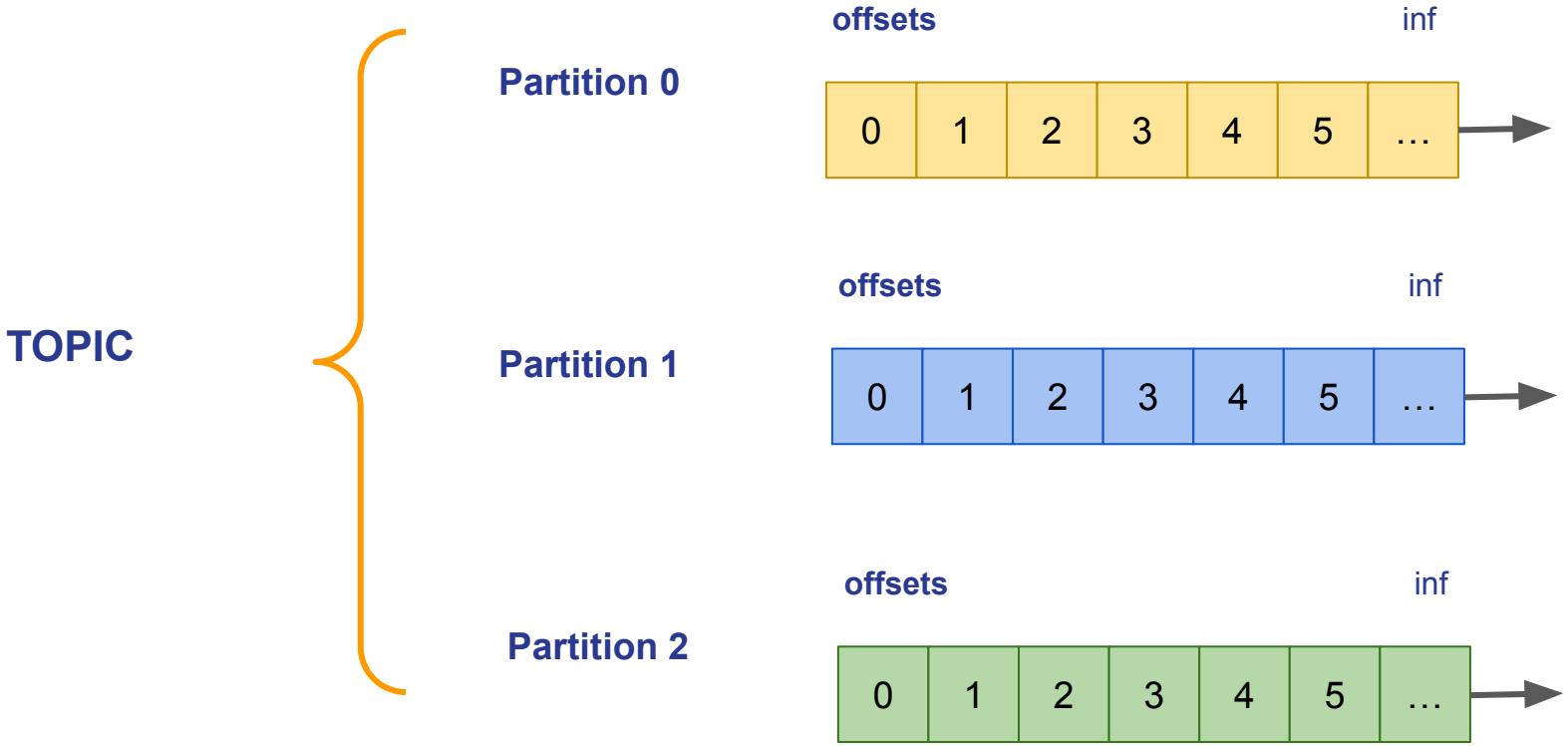


# Topics

- Similar a una **tabla** en BD
- Puedes tener cuantos **topics** se deseen
- Se identifica por su **nombre**
- Están divididos en **particiones**



# Tópico, particiones y offsets



# Table of Contents


**Recap**
Resumen de conceptos base de Kafka

**Topics & Brokers**
Manejo de un tópico dentro de Kafka

**Producers**
Función de un productor en Kafka

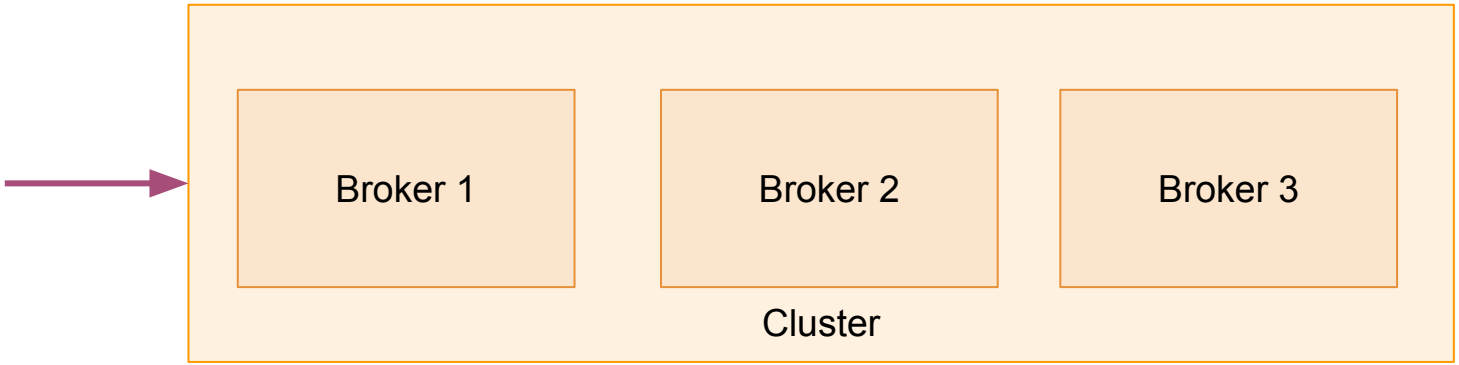
**Consumers**
Función de un consumidor en Kafka

**Consumer Group & Offsets**
¿Qué pasa cuando agrupo a mis consumidores?

# Topics y Brokers

# Brokers

- Un **cluster de kafka** está compuesto de **múltiples brokers** (servers)
- Cada broker solo contiene **ciertas particiones** del tópico (pero no todas)
- Básicamente cada **broker tiene datos** pero **no** contienen **todos los datos**
- Al **conectarte** a cualquier **broker**, te **conectas** a todo el **cluster de brokers** de kafka



**TOPIC-A**

Partition 0

offsets inf

| 0 | 1 | 2 | 3 | 4 | 5 | … |
|---|---|---|---|---|---|---|

Partition 1

offsets inf

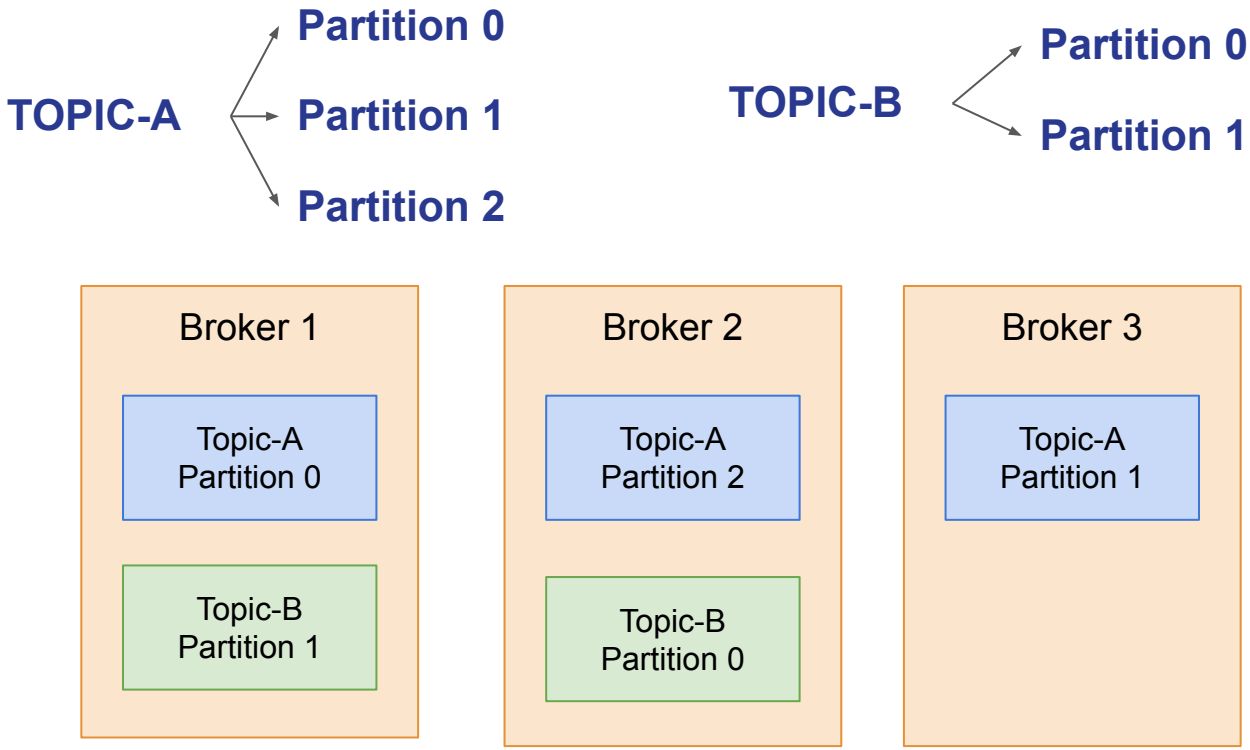
| 0 | 1 | 2 | 3 | 4 | 5 | … |
|---|---|---|---|---|---|---|

Partition 2

offsets inf

| 0 | 1 | 2 | 3 | 4 | 5 | … |
|---|---|---|---|---|---|---|

# Ejemplo



# Time for a Hands-On Exercise

# Table of Contents


**Recap**
Resumen de conceptos base de Kafka

**Topics & Brokers**
Manejo de un tópico dentro de Kafka

**Producers**
Función de un productor en Kafka

**Consumers**
Función de un consumidor en Kafka

**Consumer Group & Offsets**
¿Qué pasa cuando agrupo a mis consumidores?

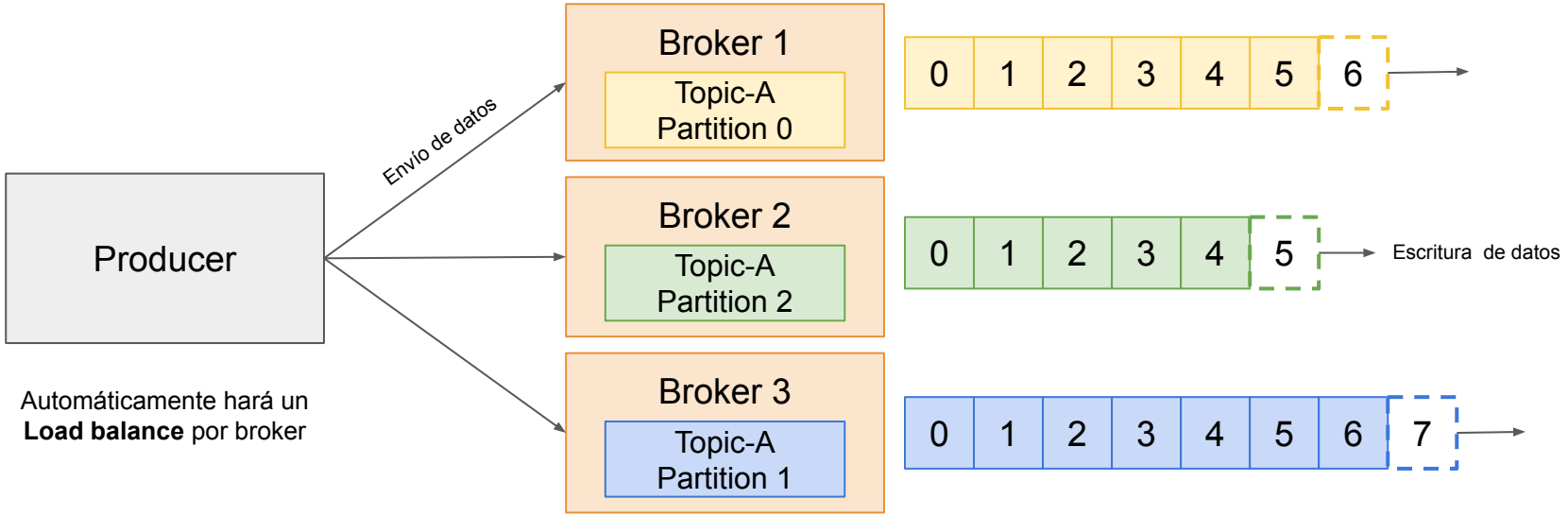
# Break Time

10 min

# Producers

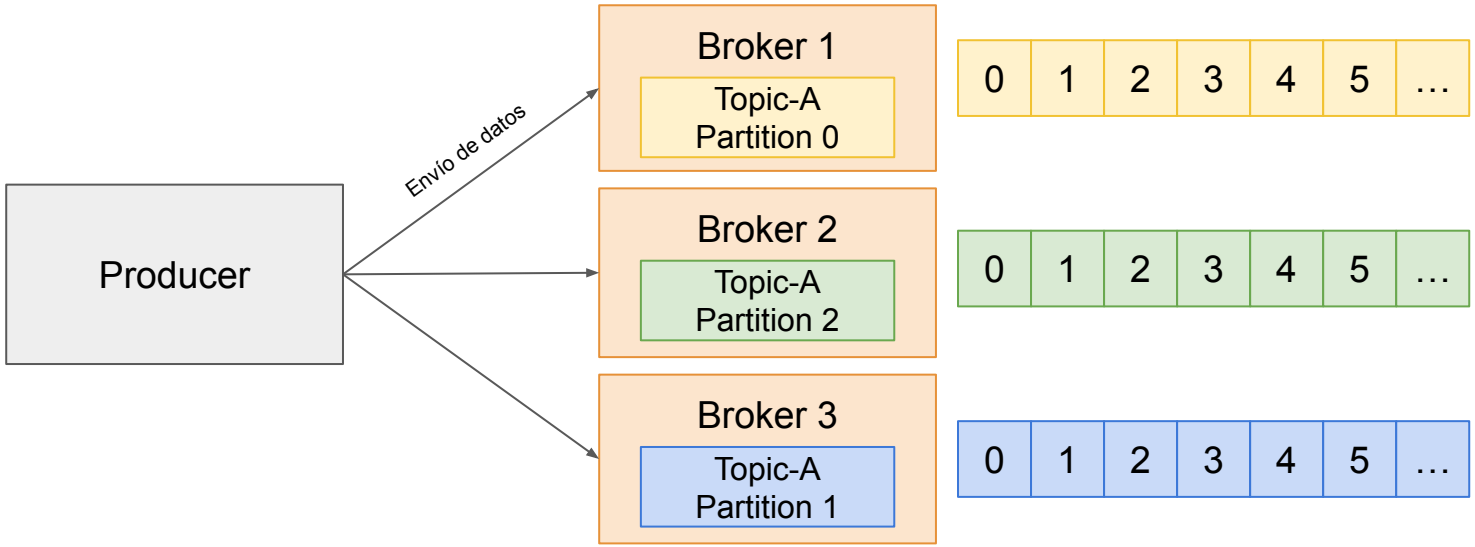
# Producers

- **Escriben datos** a los **tópicos**
- Automáticamente saben a qué **broker** y **partición tienen** que **escribir**
- En caso de alguna **falla** en los brokers, los productores sabrán **automáticamente** hará un **recover**



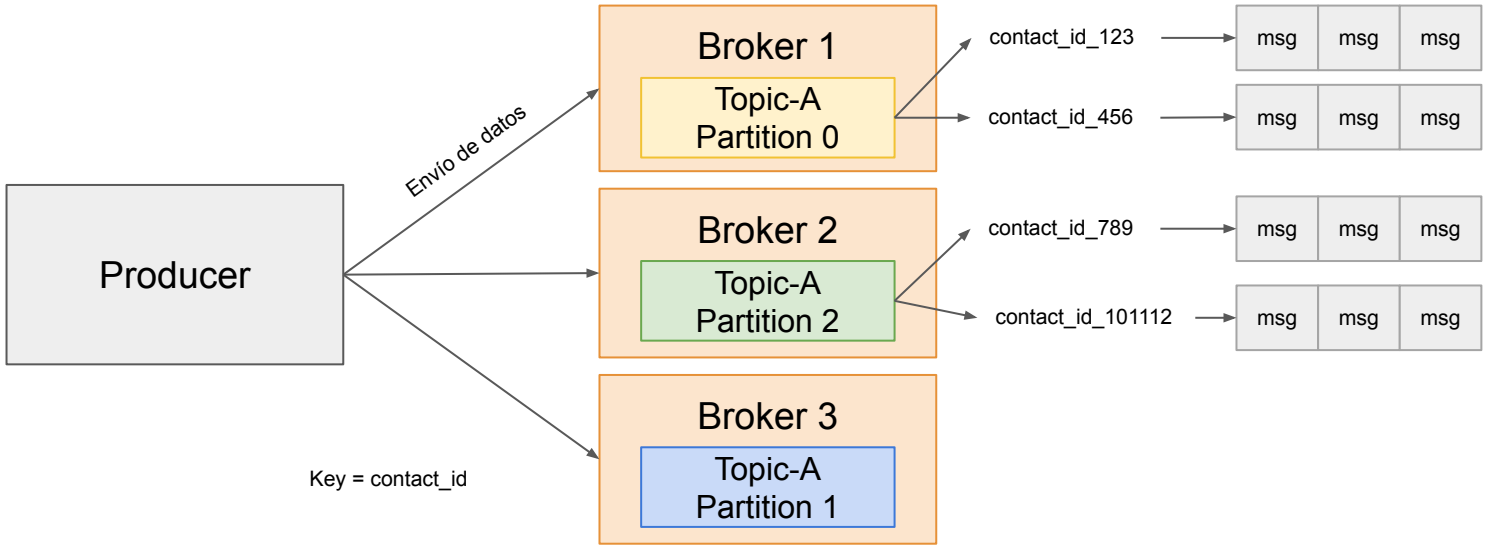
# Producers

- Se puede configurar para recibir el reconocimiento (acknowledgement) de que se escribieron los datos
  - **Acks = 0** : El productor **no esperará** por el **reconocimiento** (posible **pérdida de datos**)
  - **Acks = 1** : (default) El productor **esperará** por el **reconocimiento** (la **pérdida** de datos es **limitada**)
  - **Acks = all**: El productor esperará por el reconocimiento seguro (tanto en el broker destino como en las réplicas por tolerancia por fallo (**sin pérdida de datos**)



# Producers: Messages Keys

- Los productores pueden enviar una llave (**key**) en el mensaje (string, number, etc)
- **Key = null**, los datos son enviados por **round robin** (**bkr 1,** luego **bkr 2** y luego **bkr 3** ...)
- **Key != null**, todos los mensajes con la misma llave se enviarán siempre a la misma partición.
- Una **key** básicamente es algo que especificamos que requiere el mensaje para **ordenar** de acuerdo un campo específico (ej. User_id, contact_id, timestamp, date, etc)



# Time for a Hands-On Exercise

# Table of Contents


**Recap**

Resumen de conceptos base de Kafka

**Topics & Brokers**

Manejo de un tópico dentro de Kafka

**Producers**

Función de un productor en Kafka

**Consumers**

Función de un consumidor en Kafka

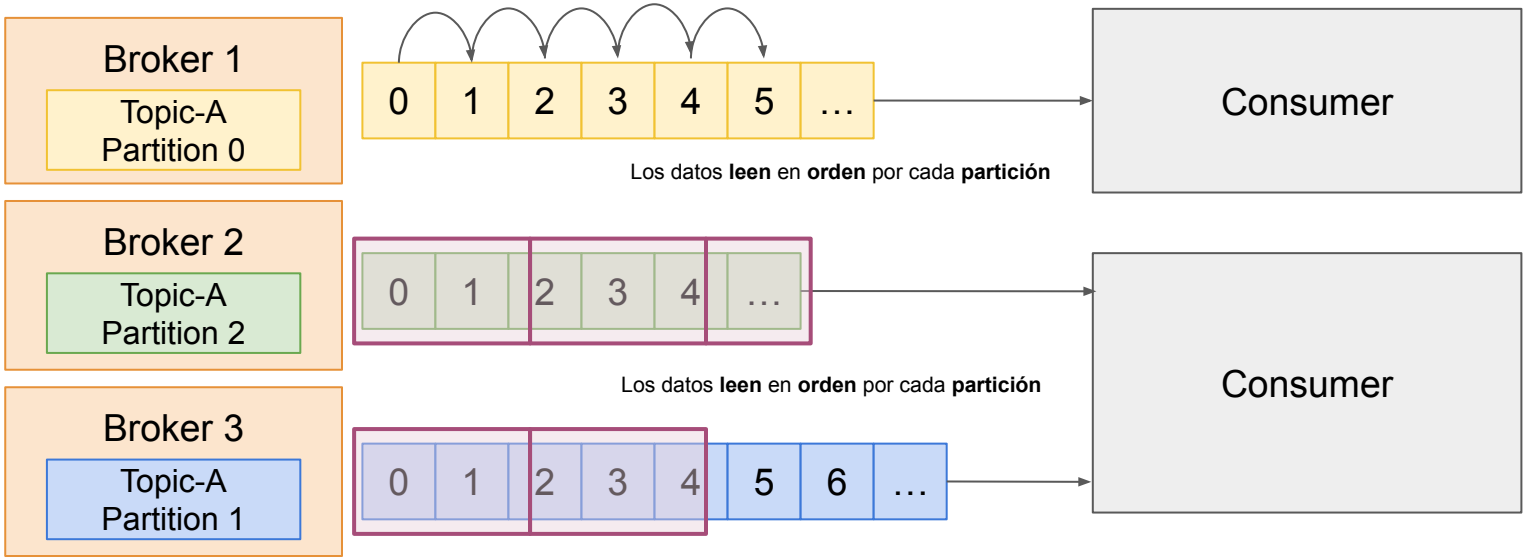
**Consumer Group & Offsets**

¿Qué pasa cuando agrupo a mis consumidores?

# Consumers

# Consumers

- **Leen** los **datos** de un tópico (identificado por su nombre)
- Los consumidores saben desde qué broker deben leer
- En **caso de falla** con el **broker**, los consumidores **saben** como **hacer** un **recover**
- Los datos se **leen** en **orden** por **cada partición**



# Time for a Hands-On Exercise

# Table of Contents


**Recap**

Resumen de conceptos base de Kafka

**Topics & Brokers**

Manejo de un tópico dentro de Kafka

**Producers**

Función de un productor en Kafka

**Consumers**

Función de un consumidor en Kafka

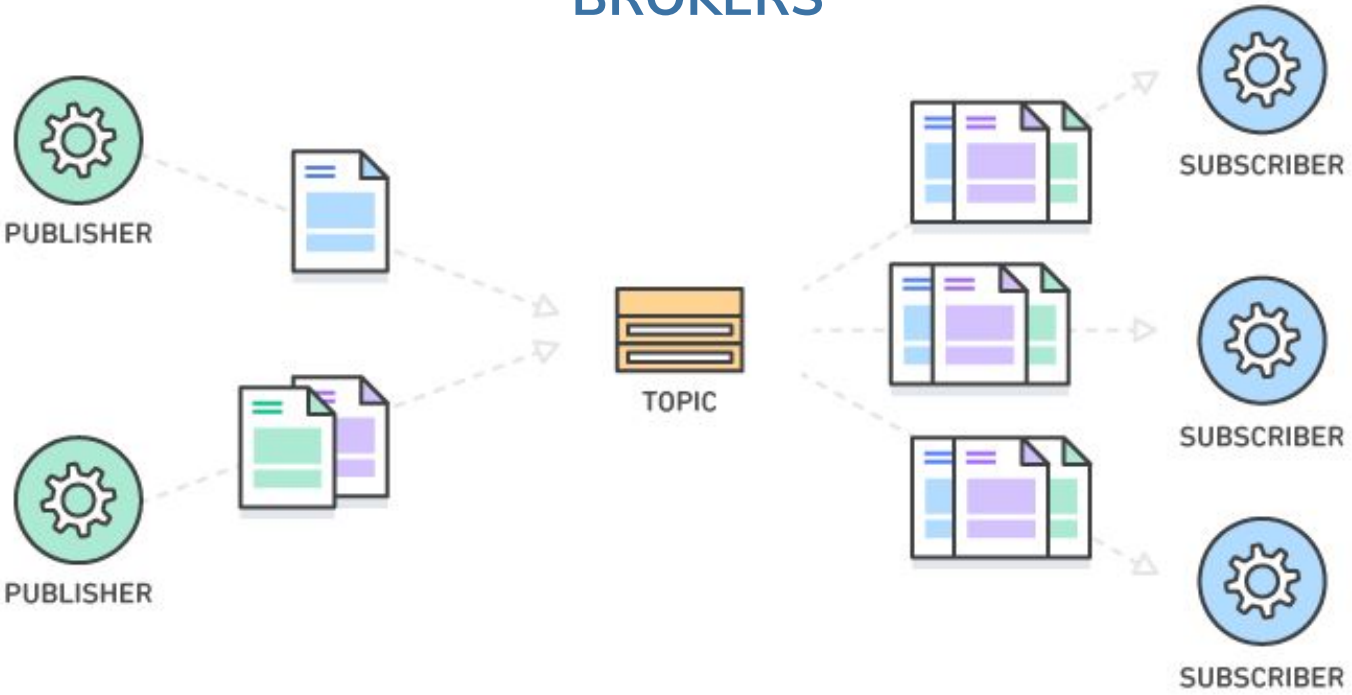
**Consumer Group & Offsets**

¿Qué pasa cuando agrupo a mis consumidores?

## PRODUCERS

## TOPICS & BROKERS

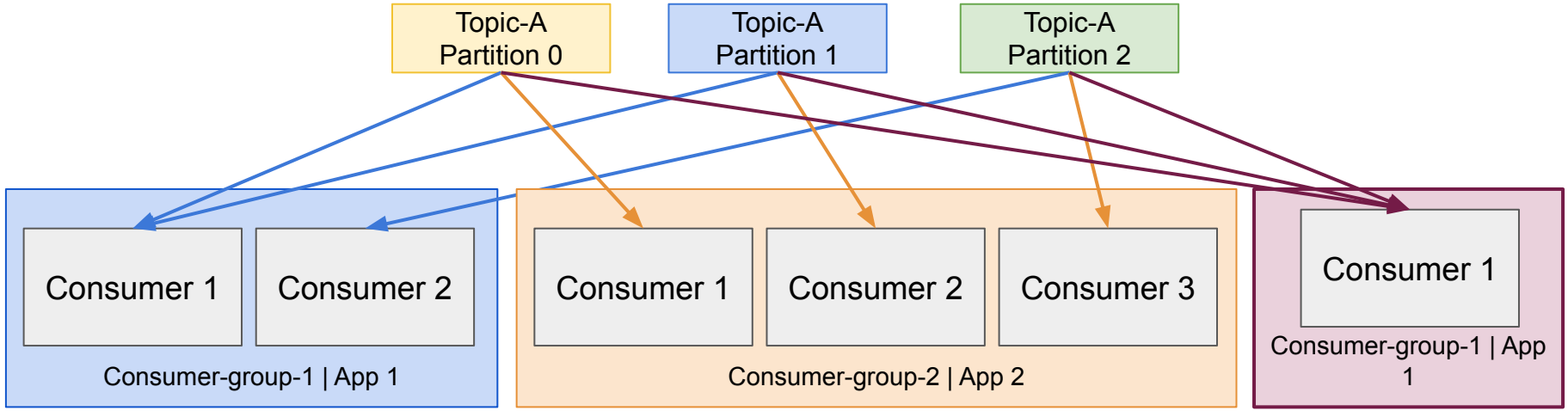
## CONSUMERS



# Consumers Group & Offsets

# Consumers Groups

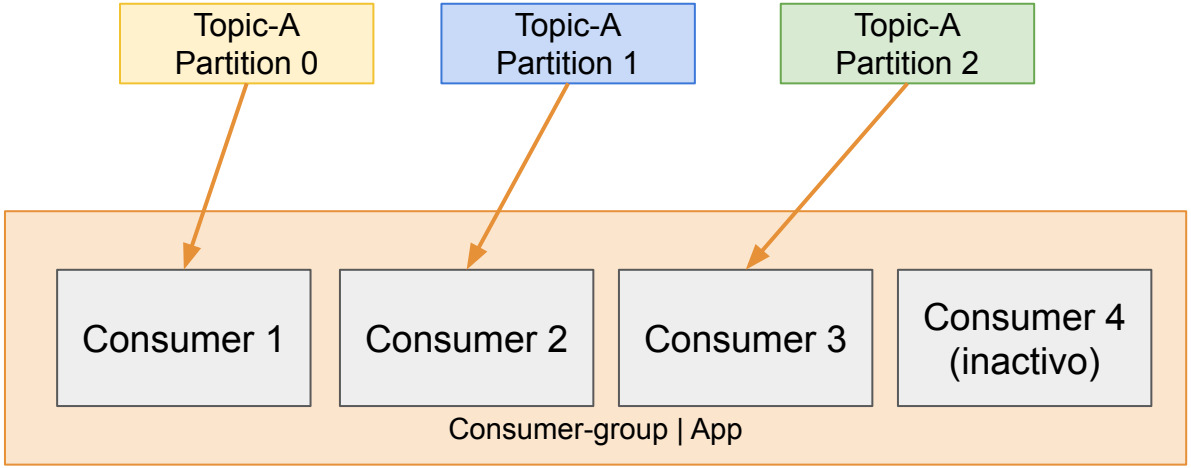
- Cada consumidor dentro del grupo lee exclusivamente una partición
- Si hay más consumidores que particiones, entonces habrá consumidores inactivos




Nota: los consumidores automáticamente usaran el **GroupCoordinator** y **ConsumerCoordinator** para asignar consumidores a una partición

# ¿Qué pasa si tenemos demasiados consumidores?

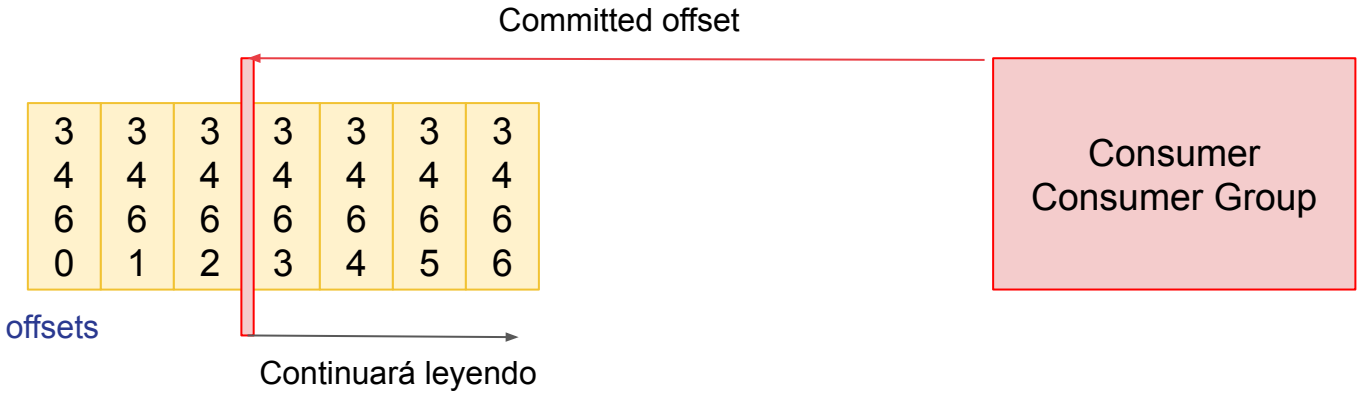
- Si hay más consumidores que particiones, algunos consumidores estarán inactivos



# Consumer Offsets

- Kafka guarda los offsets en donde el consumer group ha sido leído
- Los offsets **asignados/committed** viven dentro de un tópico de kafka (bajo el nombre **__consumer_offsets**)
- Cuando un consumidor en un grupo tiene que procesar datos recibidos desde kafka, lo que hará será escribir estos offsets a el tópico __consumer_offsets (se hace automático)

- ¿Por qué se hace esto?
- Porque si el consumidor muere o falla, y el consumidor se reinicia o se tiene un backup, este podrá reiniciar la lectura desde donde se dejó el punto muerto gracias a que se asignó el offset



# Delivery Semantics para los consumidores

- Consumidores **eligen cuándo** hacer **commit** de los offsets
- Existen **3 tipos** de semánticas para entregas:
    - **At most once**
        - Los offsets hacen committed tan pronto se reciben
        - Si falla algo, el mensaje se perderá (no se leerá de nuevo)
    - **At least one** (usualmente preferido)
        - Se hace el committed después que el mensaje es procesado
        - Si falla, el mensaje se podrá leer de nuevo
        - Puede resultar en que se duplique la lectura de mensajes. Asegurarnos que sea idempotente (ej. Si leemos de nuevo el mensaje no afecte a nuestro sistema)
    - **Exactly once**
        - Es el Santo Grial
        - Solo puede ser logrado con flujos de **kafka** a **kafka** o alguna otra herramienta que nos lo garantice
        - Pero si lo hacemos de **kafka** a un **servicio externo**, se usa un **consumidor idempotente**

# Tiempo de ejercicio

# Recap

Recapitulando...

- Describir Kafka
- ¿Para qué sirve un tópico?
- Diferencia entre Productor y Consumidor
- Broker vs Cluster

## Practice Exercise

1. Generar un topic con más de 2 particiones
2. Generar el Productor en Java y agrupar en llaves 10msj dependiendo si son números pares o ímpares
3. Generar un consumidor en Java e imprimir la lista de pares e impares que recibimos del productor, con los datos de la llave y partición

- Entregar impresión de pantalla de la consola y la terminal de productor y consumidor
- Impresión de pantalla con el - -describe del tópico mostrando las particiones

# Q&A

## Feedback Form

Let us know your feedback!

https://forms.gle/WKtc8wZeSxWnjGo8A

# Thank you