TAREA 2: SISTEMAS DISTRIBUIDOS
Kafka: eventos
Profesor: Nicolás Hidalgo
Ayudantes: Cristian Villavicencio, Joaquín Fernandez, Nicolás Núñez y Felipe Ulloa

LEA EL DOCUMENTO COMPLETO ANTES DE EMPEZAR A DESARROLLAR LA TAREA

Objetivo

El objetivo de este trabajo es introducir a los estudiantes a los *sistemas de procesamientos basados en eventos de tipo stream*. Para ello, los alumnos deberán trabajar con **Apache Kafka** un sistema de *micro-batching* ampliamente utilizando en la industria hoy en día. Los estudiantes deberán comprender y aplicar las funcionalidades de Kafka, configurar sus componentes y reconocer las ventajas de este tipo de tecnologías.

Conceptos previos

Apache Kafka es una plataforma distribuida de transmisión de datos que permite *publicar*, *almacenar* y *procesar* flujos de registros, así como *suscribirse* a ellos, de forma inmediata. Está diseñada para administrar los flujos de datos de varias fuentes y distribuirlos a diversos usuarios.

Kafka posee algunos términos que debe conocer:

- **Broker**: Corresponde a un servidor de Kafka. Pueden existir varios conectados en una red y son capaces de comunicarse entre ellos utilizando un mecanismo propio. Siempre existe un broker líder; en caso de que este se caiga, otro entra al mando.
- **Producer**: Es aquel que produce/publica datos en algún flujo.
- **Topic**: Es un canal donde se publica el flujo de datos. Puede relacionarse con una estructura de datos como lo es una cola.
- **Consumer**: Es aquel que consume los datos publicados en un topic. Sin embargo, los datos al ser consumidos **no** son borrados del topic.
- **Partition**: Es parte del flujo generado en un topic. Podría decirse que son *mini-topics* y pueden repartirse entre distintos brokers. Cada partición puede contener información distinta, sin embargo, se utilizan para ordenar una jerarquía con la información.
- **Consumer group**: Dado que los datos de un topic de Kafka no son borrados, existe este mecanismo para guardar el *offset* de la lectura de los datos. Es decir, que si un consumer group lee la información del consumer group X, Kafka se encargará de guardar la posición del último valor leído, para así no perder el orden. Dos *consumer groups* distintos tendrán distintos *offset*.

Para más detalles, revisar la documentación oficial de apache Kafka: `https://kafka.apache.org/`. Por otro lado, se le recomienda revisar el siguiente repositorio: `https://github.com/Naikelin/async-events-kafka`

Problemática

El gremio de *Maestros Motehuesilleros* de Chile *MAMOCHI*, encargado de establecer políticas legales para la venta en carritos de **Mote con Huesillo**, ha crecido a un ritmo agigantado. Los anticuados métodos de trabajo para dar soporte a las tareas del gremio requiere de una actualización que implica la utilización de plataformas informáticas capaces de gestionar dichos procesos de la manera más eficiente y escalable.

En primer lugar, un *maestro Motehuesillero*, para poder ser miembro de *MAMOCHI*, debe mandar un formulario de inscripción escrito a mano. Este proceso se realiza a través de correo (a la antigua, con una carta!). De la misma manera, si por alguna razón se desea apresurar el proceso, uno puede realizar un pago para que el gremio revise la inscripción de manera prioritaria.

Por otro lado, las ventajas de ser parte del gremio, corresponden a:

- Tener ayuda en la gestión de los ingredientes del Mote con Huesillo. Esto implica que si se te acaba el stock, puedes avisar (llamando por teléfono) y los distribuidores repondrán los insumos para el carrito de manera inmediata.
- Tener un recuento de ventas semanales hechos a mano por los encargados del gremio. Este se envía como Carta a la casa del *Maestro Motehuesillero*, luego de que el gremio se encargase de contar las boletas por usted.

Como se puede apreciar, estos procesos manuales son gestionados por gente que ya no está dando abasto. Pues, hoy en día, debido al calentamiento global, nunca está de más un Mote con Huesillo cuando hay más de 35°C en las ciudades de Chile. Es así como *MAMOCHI* lo contactó a usted para que desarrolle un sistema distribuido y escalable, para así automatizar estos procesos y aliviar la carga de los funcionarios del gremio.

Solución

Para resolver la problemática del gremio de *Maestros Motehuesilleros*, se propone el desarrollo de un sistema distribuido basado en ***Apache Kafka*** que permitirá automatizar y optimizar los procesos de inscripción, gestión de ingredientes, y recuento de ventas.

Arquitectura del Sistema

El sistema propuesto consta de 3 componentes: Productores, Consumidores y Tópicos. A continuación se detallan las funciones de cada componente.

Productores Kafka:

- **Formulario de Inscripción**: Un servicio web que permitirá a los maestros Motehuesilleros enviar sus formularios de inscripción de manera digital. De alguna manera, si el mensaje está marcado como **Paid**, se debe enviar a través de una cola especial *Partición aparte del tópico correspondiente*. Cuando el formulario esté procesado, se les enviará por *Email*, las credenciales (de la API en este caso) para que el Maestro se identifique y pueda hacer valer sus beneficios.
- **Gestión de Ingredientes**: Un sistema que permitirá a los maestros notificar automáticamente cuando se les acabe el stock. Considere que los mismos maestros notificarán la falta de stock por medio de un sitio web o aplicación ad-hoc.
- **Recuento de Ventas**: Un servicio que registrará las ventas realizadas por cada maestro para facilitar la contabilidad.

Tópicos Kafka:

- Frente a los requerimientos entregados, usted deberá modelar los tópicos de Kafka.

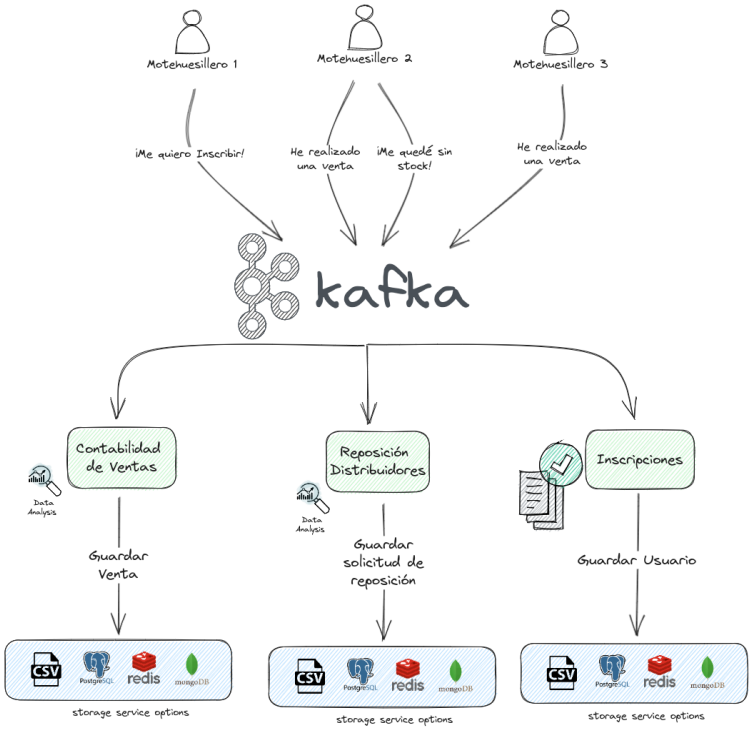
Consumidores Kafka:

- **Proceso de Aprobación**: Un servicio que consumirá los mensajes relacionados con las inscripciones y gestionará el proceso de aprobación.

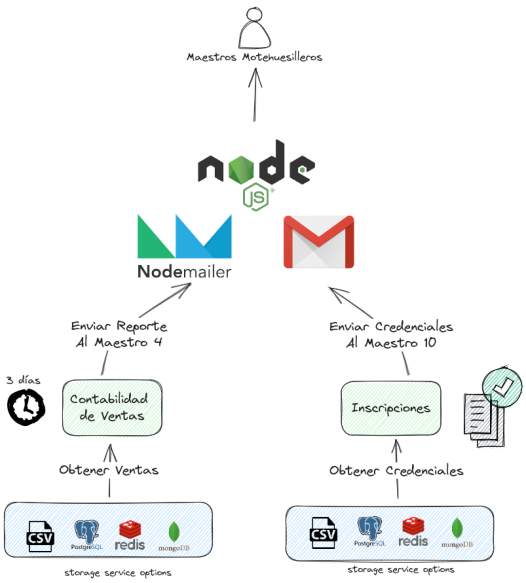
- **Distribuidores**: Un servicio que consumirá los mensajes relacionados con la reposición de stock y coordinará la reposición de stock. Puede asumir que cuando se emite y procesa este evento, el stock del carrito aumenta inmediatamente.
- **Contabilidad**: Un servicio que consumirá los mensajes relacionados con las ventas y realizará el recuento de ventas semanal. Al terminar la semana se enviará un correo mostrando las estadísticas:

  1. Cantidad de ventas realizadas.
  2. Ganancias.

  A continuación se les presenta un diagrama básico de la arquitectura:



Además, se presenta un diagrama de flujo para el envío de correo al Maestro Mote Huesillero, en el cual se incluye una librería de ejemplo para facilitar el envío de correo:



Instrucciones

- Estudiar y comprender el funcionamiento de Kafka: Conceptos básicos y la implementación frente al lenguaje que usted utilice.
- Definir un escenario de evaluación, el cual será el a implementar y presentar más adelante. Un ejemplo de escenario podría ser:
  - **Maestros a inscribir**: 5 maestros, de los cuales 2 serán **Paid**.
  - **Periodo de ventas por maestro**: Cada maestro vende en un periodo de entre 5-7 s.
  - **Cada cuántas ventas a un maestro se le acaba el stock**: Cada 10 ventas realizadas, el maestro se queda sin stock.
- Implementar el sistema con la arquitectura propuesta: Se debe mantener la idea de la arquitectura utilizando Kafka. Mientras que en los servicios que se desarrollen (Proceso de Aprobación, Distribuidores y Contabilidad) puede utilizar herramientas externas. Por ejemplo una base de datos local (SQL, MongoDB, Redis, etc...).
- Una vez implementado el sistema, responder las siguientes preguntas:
  1. Describa en detalle cómo la arquitectura de Apache Kafka permite la tolerancia a fallos del sistema propuesto.
  2. Considerando el sistema propuesto, ¿cómo se aseguraría de que el sistema es capaz de escalar para manejar un aumento significativo en el número de *Maestros Motehuesilleros* y en la cantidad de transacciones por segundo? Describa cualquier ajuste de configuración o estrategia de escalado que implementaría.
  3. Utilizando métricas empíricas, evalúe el rendimiento del sistema bajo diferentes cargas de trabajo. ¿Cómo afecta el aumento de la cantidad de mensajes en los tópicos a la latencia y al throughput del sistema? Presente gráficos o tablas que respalden su respuesta. Se recomienda que se realicen pruebas automatizadas con scripts.
  4. Dada la naturaleza variable de la demanda de *Mote con Huesillo*, ¿cómo podría optimizar el uso de recursos del sistema durante los períodos de baja demanda, y cómo prepararía el sistema para los picos de demanda? Explique las estrategias de optimización y autoscaling que podría emplear.
  5. Considerando la diversidad de fuentes de datos en el sistema propuesto (formularios de inscripción, gestión de ingredientes, recuento de ventas), ¿cómo se aseguraría de que los datos se integran y gestionan de manera coherente y unificada en Apache Kafka? Describa los posibles desafíos y soluciones para la integración de datos y la gestión de esquemas en los tópicos de Kafka.

Entrega

Para la entrega de esta tarea, usted deberá realizar:

- Un repositorio con todos los códigos utilizados.
- Un video donde solamente se muestre el funcionamiento del sistema.
- Un informe donde explique, con sus palabras, brevemente el código desarrollado. Se deben tener las secciones: *Problema y solución*, *Explicación de módulos de código*, *Configuración de Kafka* y *Respuestas a las preguntas*.

Aspectos formales de entrega

- **Fecha de entrega:** 31 de Octubre 23:59 hrs.
- **Número de integrantes:** Grupos de 2 personas las cuales deben estar claramente identificadas en la tarea.
- **Pauta de evaluación:** La pauta se puede encontrar en el siguiente link: `https://docs.google.com/spreadsheets/d/1dr4RpPx6RtCE0tbOYhQVk2qn292Hs5mI-10tqunPobM/edit?usp=sharing`.
- **Lenguaje de Programación:** para la implementación debe escoger entre los siguientes lenguajes: **Python**, **Java**, **Go** o **JavaScript**.
- **Formato de entrega:** Repositorio público (Github o Gitlab), video de funcionamiento e informe en formato PDF.
- **Tecnologías complementarias:** En caso de usar tecnologías complementarias, añadir una descripción en el informe.
- **Contenedores de Kafka**: se recomienda utilizar las siguientes imágenes: `https://hub.docker.com/r/bitnami/kafka/`. En caso de utilizar otra imagen, deberá especificarlo en el informe.
- Las copias de código serán penalizadas con nota mínima. Referente apropiadamente todo segmento de código que no sea de su autoría.
- No se debe implementar un front o interfaz, solo basta con implementar una API REST o una interfaz interactiva en la terminal.
- Consultas: **nicolas.nunez2@mail.udp.cl** o **naike_1** — **felipe.ulloa1@mail.udp.cl** o **felipe_ulloa**