

Actividad 5 – Módulo 8

1. Instalación de Kafka en Windows

a. Descargamos Kafka: <https://kafka.apache.org/downloads>

Para este ejercicio utilizamos la versión 3.9.1

b. Iniciamos Zookeeper en una terminal de powershell

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd C:\kafka
> bin\windows\zookeeper-server-start.bat config\zookeeper.properties
[2025-09-24 20:13:23,723] INFO Reading configuration from: config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,723] WARN config\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,731] WARN tmp\zookeeper is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,733] INFO ClientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,733] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,733] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,733] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,737] INFO autopurge.snapshotCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2025-09-24 20:13:23,737] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2025-09-24 20:13:23,737] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2025-09-24 20:13:23,737] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2025-09-24 20:13:23,741] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2025-09-24 20:13:23,741] INFO Reading configuration from: config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,741] WARN config\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-24 20:13:23,741] WARN tmp\zookeeper is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

c. Iniciamos Kafka server en una terminal de powershell

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd C:\kafka
> bin\windows\kafka-server-start.bat .\config\server.properties
[2025-09-24 20:20:38,322] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration)
[2025-09-24 20:20:38,582] INFO Setting -Djdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2025-09-24 20:20:38,662] INFO starting (kafka.server.KafkaServer)
[2025-09-24 20:20:38,662] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2025-09-24 20:20:38,702] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2025-09-24 20:20:38,728] INFO Client environment:zookeeper.version=3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.ZooKeeper)
[2025-09-24 20:20:38,728] INFO Client environment:host.name=192.168.1.13 (org.apache.zookeeper.ZooKeeper)
[2025-09-24 20:20:38,728] INFO Client environment:java.version=1.8.0_202 (org.apache.zookeeper.ZooKeeper)
[2025-09-24 20:20:38,728] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2025-09-24 20:20:38,728] INFO Client environment:java.home=C:\Program Files\Java\jdk1.8.0_202\jre (org.apache.zookeeper.ZooKeeper)
[2025-09-24 20:20:38,728] INFO Client environment:java.class.path=C:\kafka\libs\activation-1.1.1.jar;C:\kafka\libs\aalpalliance-repackaged-2.6.1.jar;C:\kafka\libs\argparse4j-0.7.0.jar;C:\kafka\libs\audience-annotations-0.12.0.jar;C:\kafka\libs\caffeine-2.9.3.jar;C:\kafka\libs\commons-beanutils-1.9.4.jar;C:\kafka\libs\commons-cli-1.4.jar;C:\kafka\libs\commons-collections-3.2.2.jar;C:\kafka\libs\commons-digester-2.1.jar;C:\kafka\libs\commons-io-2.14.0.jar;C:\kafka\libs\commons-lang3-3.12.0.jar;C:\kafka\libs\commons-logging-1.2.jar;C:\kafka\libs\commons-validator-1.7.jar;C:\kafka\libs\connect-api-3.9.1.jar;C:\kafka\libs\connect-basic-auth-extension-3.9.1.jar;C:\kafka\libs\connect-file-3.9.1.jar;C:\kafka\libs\connect-json-3.9.1.jar;C:\kafka\libs\connect-mirror-3.9.1.jar;C:\kafka\libs\connect-mirror-client-3.9.1.jar;C:\kafka\libs\connect
```

d. Creamos el Topic en una terminal de powershell

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd C:\kafka
> bin\windows\kafka-topics.bat --create --topic user-activity --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1
Error while executing topic command : Topic 'user-activity' already exists.
[2025-09-24 20:21:51,282] ERROR org.apache.kafka.common.errors.TopicExistsException: Topic 'user-activity' already exists.
(org.apache.kafka.tools.TopicCommand)
PS C:\kafka>
```

e. Verificamos que Kafka este funcionando correctamente:

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd C:\kafka
> bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092
user-activity
PS C:\kafka>
```

2. Implementación

Se adjunta al final del informe los códigos

a. Productor de Eventos (`producer.py`)

El productor simula 10 tipos diferentes de eventos de usuario:

Características implementadas:

- Simulación de 5 usuarios únicos (IDs: 1-5)
- 10 productos diferentes (Gaming, Periféricos, Accesorios)
- 3 tipos de acciones: `viewed_product`, `added_to_cart`, `purchased_product`
- Timestamps automáticos en formato ISO
- Intervalos aleatorios entre eventos (1-3 segundos)

b. Consumidor de Eventos (`consumer.py`)

El consumidor procesa eventos en tiempo real y mantiene estadísticas:

Funcionalidades implementadas:

- Base de datos SQLite integrada (`user_events.db`)
- Estadísticas en tiempo real
- Sistema de alertas inteligente
- Cálculo de tasas de conversión
- Almacenamiento persistente de eventos
- Manejo de grupos de consumidores

3. Verificación

Consola producer:

- Elementos enviados:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Evento enviado: {'user_id': 1, 'action': 'viewed_product', 'product_id': 104, 'product_name': 'Monitor 4K', 'timestamp': '2025-09-24T21:14:53.783383'}
Evento enviado: {'user_id': 5, 'action': 'viewed_product', 'product_id': 106, 'product_name': 'Silla Gaming', 'timestamp': '2025-09-24T21:14:55.238078'}
Evento enviado: {'user_id': 5, 'action': 'viewed_product', 'product_id': 106, 'product_name': 'Silla Gaming', 'timestamp': '2025-09-24T21:14:56.788464'}
Evento enviado: {'user_id': 3, 'action': 'added_to_cart', 'product_id': 107, 'product_name': 'Webcam HD', 'timestamp': '2025-09-24T21:14:59.678682'}
Evento enviado: {'user_id': 1, 'action': 'viewed_product', 'product_id': 104, 'product_name': 'Monitor 4K', 'timestamp': '2025-09-24T21:15:02.578171'}
Evento enviado: {'user_id': 2, 'action': 'added_to_cart', 'product_id': 102, 'product_name': 'Mouse Inalámbrico', 'timestamp': '2025-09-24T21:15:05.066543'}
Evento enviado: {'user_id': 1, 'action': 'viewed_product', 'product_id': 104, 'product_name': 'Monitor 4K', 'timestamp': '2025-09-24T21:15:07.940481'}
Evento enviado: {'user_id': 5, 'action': 'viewed_product', 'product_id': 106, 'product_name': 'Silla Gaming', 'timestamp': '2025-09-24T21:15:09.537659'}
Evento enviado: {'user_id': 5, 'action': 'purchased_product', 'product_id': 109, 'product_name': 'Mousepad XXL', 'timestamp': '2025-09-24T21:15:11.216705'}
Evento enviado: {'user_id': 1, 'action': 'viewed_product', 'product_id': 104, 'product_name': 'Monitor 4K', 'timestamp': '2025-09-24T21:15:12.669283'}
Evento enviado: {'user_id': 4, 'action': 'added_to_cart', 'product_id': 108, 'product_name': 'Auriculares Gaming', 'timestamp': '2025-09-24T21:15:15.668714'}
Evento enviado: {'user_id': 4, 'action': 'viewed_product', 'product_id': 108, 'product_name': 'Micrófono USB', 'timestamp': '2025-09-24T21:15:17.145586'}
Evento enviado: {'user_id': 2, 'action': 'purchased_product', 'product_id': 102, 'product_name': 'Mouse Inalámbrico', 'timestamp': '2025-09-24T21:15:18.866469'}
Evento enviado: {'user_id': 4, 'action': 'viewed_product', 'product_id': 108, 'product_name': 'Micrófono USB', 'timestamp': '2025-09-24T21:15:20.490898'}
Evento enviado: {'user_id': 3, 'action': 'purchased_product', 'product_id': 103, 'product_name': 'Teclado Mecánico', 'timestamp': '2025-09-24T21:15:21.946681'}
Evento enviado: {'user_id': 3, 'action': 'purchased_product', 'product_id': 103, 'product_name': 'Teclado Mecánico', 'timestamp': '2025-09-24T21:15:23.792693'}
Evento enviado: {'user_id': 4, 'action': 'added_to_cart', 'product_id': 105, 'product_name': 'Auriculares Gaming', 'timestamp': '2025-09-24T21:15:25.763424'}
```

Consola consumer:

- Elementos recibidos:
- Alertas cliente vip
- Estadísticas en tiempo real

```
Evento recibido: {'user_id': 4, 'action': 'viewed_product', 'product_id': 108, 'product_name': 'Micrófono USB', 'timestamp': '2025-09-24T21:15:17.145586'}
Evento recibido: {'user_id': 2, 'action': 'purchased_product', 'product_id': 102, 'product_name': 'Mouse Inalámbrico', 'timestamp': '2025-09-24T21:15:18.866469'}
USUARIO VIP: El usuario 2 ha realizado otra compra!

Evento recibido: {'user_id': 4, 'action': 'viewed_product', 'product_id': 108, 'product_name': 'Micrófono USB', 'timestamp': '2025-09-24T21:15:20.490898'}

Evento recibido: {'user_id': 3, 'action': 'purchased_product', 'product_id': 103, 'product_name': 'Teclado Mecánico', 'timestamp': '2025-09-24T21:15:21.946681'}
Evento recibido: {'user_id': 3, 'action': 'purchased_product', 'product_id': 103, 'product_name': 'Teclado Mecánico', 'timestamp': '2025-09-24T21:15:23.792693'}

=====
ESTADÍSTICAS EN TIEMPO REAL
=====
Total de eventos procesados: 1415
Vistas de productos: 612
Adiciones al carrito: 377
Compras realizadas: 426
Usuarios activos: 5
Tasa de adición al carrito: 61.6%
Tasa de conversión de compra: 113.8%
=====

Evento recibido: {'user_id': 4, 'action': 'added_to_cart', 'product_id': 105, 'product_name': 'Auriculares Gaming', 'timestamp': '2025-09-24T21:15:25.763424'}
Evento recibido: {'user_id': 5, 'action': 'purchased_product', 'product_id': 109, 'product_name': 'Mousepad XXL', 'timestamp': '2025-09-24T21:15:27.617673'}
Evento recibido: {'user_id': 2, 'action': 'purchased_product', 'product_id': 102, 'product_name': 'Mouse Inalámbrico', 'timestamp': '2025-09-24T21:15:29.669174'}
```

Los elementos son almacenados en la base de datos user_events.db utilizando la librería sqlite3.

producer.py

```
from kafka import KafkaProducer
import json
import time
import random
from datetime import datetime

# Configurar productor Kafka
producer = KafkaProducer(
    bootstrap_servers='localhost:9092',
    value_serializer=lambda v: json.dumps(v).encode('utf-8')
)

# Función para simular eventos de usuario
def simulate_user_activity():
    actions = [
        {"user_id": 1, "action": "viewed_product", "product_id": 101, "product_name": "Laptop Gaming"},
        {"user_id": 2, "action": "added_to_cart", "product_id": 102, "product_name": "Mouse Inalámbrico"},
        {"user_id": 3, "action": "purchased_product", "product_id": 103, "product_name": "Teclado Mecánico"},
        {"user_id": 1, "action": "viewed_product", "product_id": 104, "product_name": "Monitor 4K"},
        {"user_id": 4, "action": "added_to_cart", "product_id": 105, "product_name": "Auriculares Gaming"},
        {"user_id": 2, "action": "purchased_product", "product_id": 102, "product_name": "Mouse Inalámbrico"},
        {"user_id": 5, "action": "viewed_product", "product_id": 106, "product_name": "Silla Gaming"},
        {"user_id": 3, "action": "added_to_cart", "product_id": 107, "product_name": "Webcam HD"},
        {"user_id": 4, "action": "viewed_product", "product_id": 108, "product_name": "Micrófono USB"},
        {"user_id": 5, "action": "purchased_product", "product_id": 109, "product_name": "Mousepad XXL"}
    ]

    try:
        while True:
            # Seleccionar un evento aleatorio
            event = random.choice(actions)

            # Agregar timestamp
            event["timestamp"] = datetime.now().isoformat()

            # Enviar evento al tema 'user-activity'
            producer.send('user-activity', value=event)

            print(f"Evento enviado: {event}")

            # Esperar entre 1-3 segundos antes del siguiente evento
            time.sleep(random.uniform(1, 3))

    except KeyboardInterrupt:
        print("\nDeteniendo el productor...")
    finally:
        # Asegurar que todos los mensajes se envíen antes de cerrar
        producer.flush()
        producer.close()
        print("Productor cerrado correctamente.")

if __name__ == "__main__":
    print("Iniciando simulación de eventos de usuario...")
    print("Presiona Ctrl+C para detener")
    simulate_user_activity()
```

consumer.py

```
from kafka import KafkaConsumer
import json
from datetime import datetime
import sqlite3
import os

# Crear/conectar a una base de datos SQLite para almacenar eventos
def init_database():
    conn = sqlite3.connect('user_events.db')
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS user_events (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER,
            action TEXT,
            product_id INTEGER,
            product_name TEXT,
            timestamp TEXT,
            processed_at TEXT
        )
    ''')

    conn.commit()
    return conn

# Configurar consumidor Kafka
consumer = KafkaConsumer(
    'user-activity', # Nombre del tema
    bootstrap_servers='localhost:9092', # Dirección del servidor Kafka
    group_id='user-activity-consumer-group', # Grupo de consumidores
    value_deserializer=lambda x: json.loads(x.decode('utf-8')), # Deserializar JSON
    auto_offset_reset='latest' # Comenzar desde los mensajes más recientes
)

# Estadísticas en tiempo real
stats = {
    'total_events': 0,
    'views': 0,
    'cart_additions': 0,
    'purchases': 0,
    'active_users': set()
}

def process_event(event, db_conn):
    """Procesa un evento individual y actualiza estadísticas"""
    global stats

    # Actualizar estadísticas
    stats['total_events'] += 1
    stats['active_users'].add(event['user_id'])

    if event['action'] == 'viewed_product':
        stats['views'] += 1
    elif event['action'] == 'added_to_cart':
        stats['cart_additions'] += 1
    elif event['action'] == 'purchased_product':
        stats['purchases'] += 1

    # Almacenar en base de datos
    cursor = db_conn.cursor()
    cursor.execute('''
```

```

        INSERT INTO user_events (user_id, action, product_id, product_name, timestamp, processed_at)
        VALUES (?, ?, ?, ?, ?, ?)

    '''
    event['user_id'],
    event['action'],
    event['product_id'],
    event['product_name'],
    event['timestamp'],
    datetime.now().isoformat()
))
db_conn.commit()

return stats

def display_stats(stats):
    """Muestra estadísticas actualizadas"""
    print("\n" + "="*50)
    print("ESTADÍSTICAS EN TIEMPO REAL")
    print("="*50)
    print(f"Total de eventos procesados: {stats['total_events']}")
    print(f"Vistas de productos: {stats['views']}")
    print(f"Adiciones al carrito: {stats['cart_additions']}")
    print(f"Compras realizadas: {stats['purchases']}")
    print(f"Usuarios activos: {len(stats['active_users'])}")

    # Calcular tasas de conversión
    if stats['views'] > 0:
        cart_rate = (stats['cart_additions'] / stats['views']) * 100
        print(f"Tasa de adición al carrito: {cart_rate:.1f}%")

    if stats['cart_additions'] > 0:
        purchase_rate = (stats['purchases'] / stats['cart_additions']) * 100
        print(f"Tasa de conversión de compra: {purchase_rate:.1f}%")

    print("="*50)

def generate_alerts(event):
    """Genera alertas basadas en patrones de eventos"""
    # Alerta por compra de producto caro (simulación)
    if event['action'] == 'purchased_product' and event['product_id'] in [101, 104, 106]:
        print(f"ALERTA: Compra de producto premium por usuario {event['user_id']}: {event['product_name']}")

    # Alerta por usuario muy activo
    if event['user_id'] in [1, 2] and event['action'] == 'purchased_product':
        print(f"USUARIO VIP: El usuario {event['user_id']} ha realizado otra compra!")

def main():
    """Función principal del consumidor"""
    print("Iniciando consumidor de eventos de usuario...")
    print("Esperando eventos desde el tema 'user-activity'...")
    print("Presiona Ctrl+C para detener")

    # Inicializar base de datos
    db_conn = init_database()

    try:
        # Escuchar continuamente los eventos desde el tema
        for message in consumer:
            event = message.value

            # Procesar evento
            print(f"\nEvento recibido: {event}")

            # Actualizar estadísticas y almacenar en BD

```

```
        current_stats = process_event(event, db_conn)

        # Generar alertas si es necesario
        generate_alerts(event)

        # Mostrar estadísticas cada 5 eventos
        if current_stats['total_events'] % 5 == 0:
            display_stats(current_stats)

    except KeyboardInterrupt:
        print("\n\nDeteniendo el consumidor...")

    finally:
        # Cerrar conexiones
        consumer.close()
        db_conn.close()

        # Mostrar estadísticas finales
        print("\nESTADÍSTICAS FINALES:")
        display_stats(stats)
        print(f"\nEventos almacenados en: user_events.db")
        print("Consumidor cerrado correctamente.")

if __name__ == "__main__":
    main()
```