



---

# ARQUITECTURA STREAMING PARA BIG DATA CON APACHE KAFKA Y TWITTER

---

Por:

Héctor Fabio Banilat Quintero  
Daniel Alfonso Solano Velásquez

Profesor:

Edwin Nelson Montoya Munera



NOVIEMBRE DE 2023

UNIVERSIDAD EAFIT

Tópicos Especiales en Telemática (proyecto 3)

# Arquitectura Streaming para Big Data con Apache Kafka y Twitter

## Descripción:

El objetivo de este proyecto<sup>3</sup> es implementar una arquitectura de Streaming de Big Data, que permita capturar datos en tiempo real de la famosa plataforma de Twitter, almacenarlos en un Data Lake y analizarlos en tiempo real.

Para llevar a cabo esto, se usarán las siguientes herramientas:

- **Apache Kafka:** Será nuestro canalizador de datos en tiempo real.
- **AWS S3:** Para el tema del Data Lake.
- **Apache Spark Streaming:** Será nuestro procesador de flujos.

## Ahora bien, ¿Como será el flujo de datos?

El flujo de datos será el siguiente:

- Un productor de Kafka usará la API de Twitter para adquirir la información o los datos de los tweets.
- El productor se encargará de hacer públicos estos datos en un 'topic' de Kafka.
- Un consumidor de Kafka recibirá la información o los datos del 'topic' mencionado anteriormente y luego los almacenará en el Data Lake.
- Un procesador de flujos de Spark Streaming se encargará de leer la información que hay en el Data Lake se dispondrá a realizar un análisis de sentimientos.

## Herramientas:

Las herramientas que serán usadas para llevar a cabo esta arquitectura en el proyecto<sup>3</sup> serán las siguientes:

- **Apache Kafka:** Apache Kafka será el 'canalizador' de datos en tiempo real que permitirá publicar y suscribirse a mensajes de manera confiable. Se escogió debido a que es bastante escalable y tolerante a fallas, lo que a nuestro punto de vista, lo hace muy ideal para este tipo de aplicaciones de streaming.
- **AWS S3:** AWS S3 será nuestro servicio de almacenamiento de objetos, el cual nos ofrece un rendimiento y una escalabilidad bastante buena. Se escogió debido a que es una excelente opción para el almacenamiento de datos de streaming, permitiéndonos entrar o acceder de una forma mucho más sencilla y rápida.
- **Apache Spark Streaming:** Apache Spark Streaming será nuestro framework de procesamiento de flujos distribuido, el cual nos permitirá procesar cientos de miles de datos en tiempo real. Y después de todo, esta es una gran herramienta que sirve mucho para aplicaciones de streaming, sin dejar a un lado el tema del análisis de sentimientos.

## Otra información:

- El análisis de sentimientos se realizará utilizando un modelo de aprendizaje automático entrenado para identificar los sentimientos positivos, negativos e neutros de los tweets, puesto que contamos con conocimientos previos sobre Inteligencia Artificial.
- Una vez corriendo el análisis de sentimientos tendremos los suficientes datos para sacar provecho de ellos, tales como serían temas de creación de campañas de marketing, detección de tendencias, entre muchas otras cosas.
- Idea tomada de: <https://medium.com/@lorenagongang/sentiment-analysis-on-streaming-twitter-datausing-kafka-spark-structured-streaming-python-part-b27aecca697a>

# Proyecto No 3.

| Información | |

| --- | --- |

| Materia | Tópicos especiales en Telemática |

| Curso | ST0263 |

| Estudiantes | Hector Fabio Banilat Quintero (hfbanilatq@eafit.edu.co) |

| | Daniel Solano Velasquez (dsolano@eafit.edu.co) |

| Profesor | Edwin Nelson Montoya Munera (emontoya@eafit.edu.co) |

# 1. Objetivo

---

El objetivo de este proyecto<sup>3</sup> es implementar una arquitectura de Streaming de Big Data, que permita capturar datos en tiempo real de la famosa plataforma de Twitter, almacenarlos en un Data Lake y analizarlos en tiempo real; dichos datos deberán de ser llevados a un bucket de s3 y a un procesador de flujos que analice los datos conforme van llegando.

# 2. Video sustentación

[https://youtu.be/nybIcgaGzqU](https://youtu.be/nybIcgaGzqU)

### # 3. Aspectos solucionados y no solucionados

---

- [] Captura de datos en tiempo real. (Se creo un faker de tweets debido a que el API de twitter no estaba funcionando)
- [x] Subir los datos capturados a un Topic en Kafka.
- [x] Almacenamiento de datos en un bucket de S3.
- [x] Procesamiento de flujo de datos.
- [x] Visualización en tiempo real.

### # 4. Información general del diseño

Se empleó databricks para la creación del cluster de spark, además se empleó un VM en GCP donde se instaló kafka. Se emplea un bucket de S3 para el almacenamiento de los datos ingresados en la cola

#### ## Sobre los servicios

En esta sección, se explican los servicios implementados para dar solución al reto propuesto, explicado en la sección: 1. Objetivo.

Nombre del servicio   Rol que desempeña   IP y puertos de escucha
---

---   ---   ---
-----------------

| spark-streaming-twitter\_2.12| Libreria de spark par ala integración de la API de tweeter|

| Apache Kafka | Almacena los datos crudos consultados por el servicio de TradeStreaming. |  
34.123.51.250:9092 |

| DataBricks | Lee, procesa, transforma y carga a S3 los datos crudos, presentes en un tópic de Apache  
Kafka. | - |

| S3 | Almacena los datos transformados por el servicio de DataBricks. | s3://hfbaniatqsparks3/dataspark |

| Grafana | Permite la visualización de los datos presentes en MongoDB. | - |

## # 5. Ambiente de desarrollo

Se emplea dos notebook en databricks, con el lenguaje scala

### ## Estructura del código

.

|— databricks

| |— FakerTweetConsumer.scala

| |— FakerTweetConsumerWithStorage.scala

| |— FakerTweetProducer.scala

### ## Creacion de cuenta en databricks y creacion de cluster

1. Ve a la consola de GCP y en el buscador por Databricks
2. En los resultados selecciona el de la marketplace
3. Dale clic en solicitar, llena los datos y espera que se te active
4. Inicia sesion en databricks
5. Ve a tus workspaces y crea un nuevo workspace

6. selecciona el workspace creado, ve al menu ->new-> cluster, crea el cluster y después importa el código de los archivos .scala en la carpeta databricks.

Nota: puedes seguir este tutorial: <https://docs.gcp.databricks.com/getting-started/index.html>

# Instalacion de kafka en una instancia de GCP

```
```bash
```

# Actualiza los paquetes existentes

```
sudo apt-get update
```

# Instala Java (Kafka requiere Java para funcionar)

```
sudo apt-get install default-jdk -y
```

# Descarga Kafka desde el sitio web oficial

```
wget https://archive.apache.org/dist/kafka/2.13-2.8.0/kafka_2.13-2.8.0.tgz
```

# Descomprime el archivo descargado

```
tar -xzf kafka_2.13-2.8.0.tgz
```

# Cambia al directorio de Kafka

```
cd kafka_2.13-2.8.0
```

# Inicia el servidor Zookeeper (Kafka usa Zookeeper para la gestión del cluster)

```
bin/zookeeper-server-start.sh config/zookeeper.properties &
```

```
# Inicia el servidor Kafka
```

```
bin/kafka-server-start.sh config/server.properties &
```

```
...
```

sigue este tutorial:

<https://www.datasciencecentral.com/setting-up-your-first-kafka-development-environment-in-google/>

```
## Configuración de parámetros del proyecto
```

```
# 6. Ambiente de ejecución
```

```
---
```

```
## Arquitectura general
```

En la presente sección se mostrara la arquitectura de referencia recomendada para el proyecto, la arquitectura final y los servicios de Google Cloud Platform (GCP) empleados para el despliegue eficiente y escalable.

```
## Guía de uso
```

La siguiente guía brindará los pasos a seguir para un correcto funcionamiento y ejecución del software.

### Instalando requisitos previos

Debemos instalar las librerías usando el UI de databricks en el cluster las librerías son:

edu.stanford.nlp:stanford-corenlp:4.5.5

Maven

-

org.apache.bahir:spark-streaming-twitter\_2.12:2.4.0

Maven

-

org.apache.spark:spark-streaming-kafka-0-10-assembly\_2.12:3.5.0

## Ejecución del programa

En la UI de databricks, crea un nuevo cuaderno de scala por cada archivo que hay en la carpeta databricks,importarlos al workspace y ejecutarlos en el orde, Producer, Consumer, ConsumerWithStorage.



