TEC | Tecnológico de Costa Rica

Especialización en Ciencia de Datos

Curso de Big Data

Proyecto:

“Arquitectura de Streaming para Twitter”

Profesor:

M.Sc. Felipe Meza Obando

Estudiantes:

Claudia Alcazar Urena

Natalia Rojas Canales

Domingo 9 de Febrero, 2020

Indice

Introducción

El presente documento es producto del trabajo realizado para cumplir con el proyecto final del curso de BigData, el cual consiste en:

> Investigar y construir una aplicación básica de streaming a través de un sistema que sea capaz de identificar hashtags en publicaciones en Twitter. La arquitectura usada puede ser la que consideren apropiada, una de las más populares se planteó en la ultima clase virtual sobre el tema de streaming, se recomienda seguir la siguiente estructura:
>
> 1. Investigar arquitecturas y los bloques necesarios para lograr esta tarea, familiarización con los detalles involucrados en cada bloque, por ejemplo, uso de API de twitter etc.
> 2. Investigar como es el desarrollo de los bloques funcionales y comprensión de uso de cada uno, por ejemplo, como generar tokens/keys, uso de la interface de app en twitter etc.
> 3. Creación de un modelo básico/simple que lea streams en linea de Twitter, usado Python y Spark y que como mínimo genere una tabla con el resumen de hashtags identificados (tendencias), según el criterio definido por los integrantes del grupo.

ENTREGABLES

1- **Documento**: donde se explique la arquitectura utilizada (PDF) y uso de cada bloque (puntos 1 y 2). El presente documento.

2- **Video**: donde se lleve a cabo la demostración de sistema en operación (punto 3), se recomienda para mayor facilidad subirlo a youtube.com y proporcionar el url respectivo.
https://www.youtube.com/watch?v=Re4nj5DRaIE&feature=youtu.be

Arquitectura de Streaming para Twitter

Las nuevas tendencias tecnológicas, el procesamiento de grandes volúmenes de información y la necesidad de crear soluciones que puedan procesar y proporcionar información de valor en tiempo real, es parte de lo que las empresas viven día a día, es por ello que como parte de la investigación vamos a presentar como generar una arquitectura de procesamiento en tiempo real utilizando twitter como fuente principal.

La arquitectura propuesta consta de varios componentes tales como Kafka, responsable de orquestar la cola y almacenamiento local de los mensajes antes de ser consumidos por otra aplicación, Zookeeper, encargado de manejar la arquitectura distribuida de Kafka y la comunicación entre los componentes, Spark Streaming, es el marco de trabajo encargado de producir y consumir los mensajes, así como procesarlo para generar valor de la información recibida, HDFS, utilizado para guardar las métricas generadas y finalmente poder visualizarlas según la necesidad del cliente.

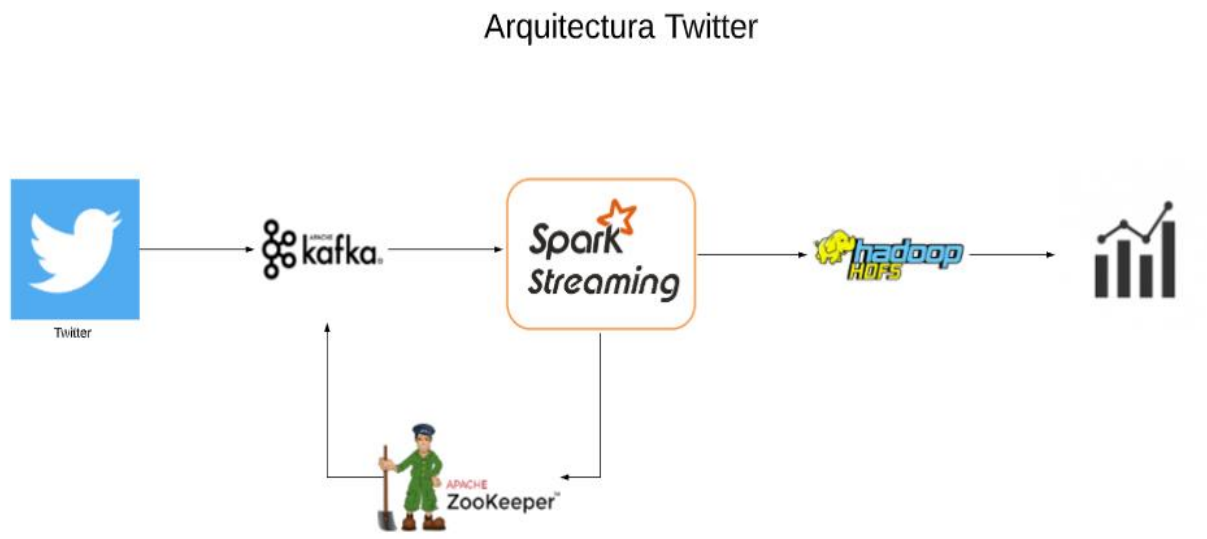

Imagen 1 – Arquitectura twitter

Fuente de Información: API de Twitter

Para crear la aplicación en Twitter primero es necesario:

Paso 1 – Crear cuenta desarrollador

Crear una cuenta para desarrollador, se realiza desde este link: https://developer.twitter.com/

Se procede a llenar un formulario que consta de 3 pasos y se debe asegurar que la información sea congruente, porque de no ser así, rechazarán la solicitud o bien, escribirán correos desde: verify@twitter.com

En nuestro caso tomo 3 días la aprobación de la misma.

Paso 2 – Crear la aplicación

Los pasos para crear la aplicación son muy sencillos>

1. Ingresa a https://developer.twitter.com/en/apps
2. Selecciona 

3. Ingresa los datos requeridos: **App name, Application description, Website URL** y **Tell us how this app will be used.**
4. Una vez creada la aplicación se procede a generar los **Keys and tokens:**

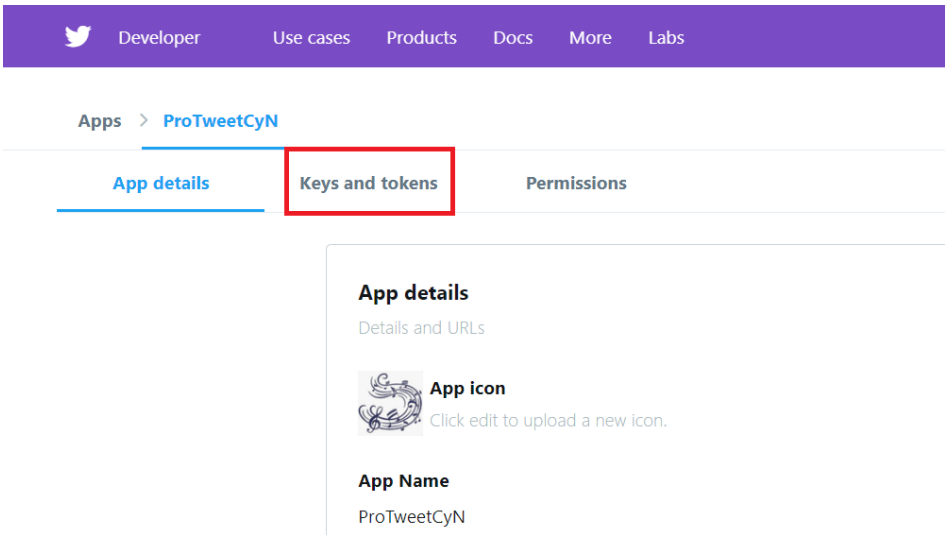

Imagen 2 –Keys and tokens

5. Ubicado en **Keys and tokens,** se utilizan los botones de 

para generar las llaves requeridas para el desarrollo.

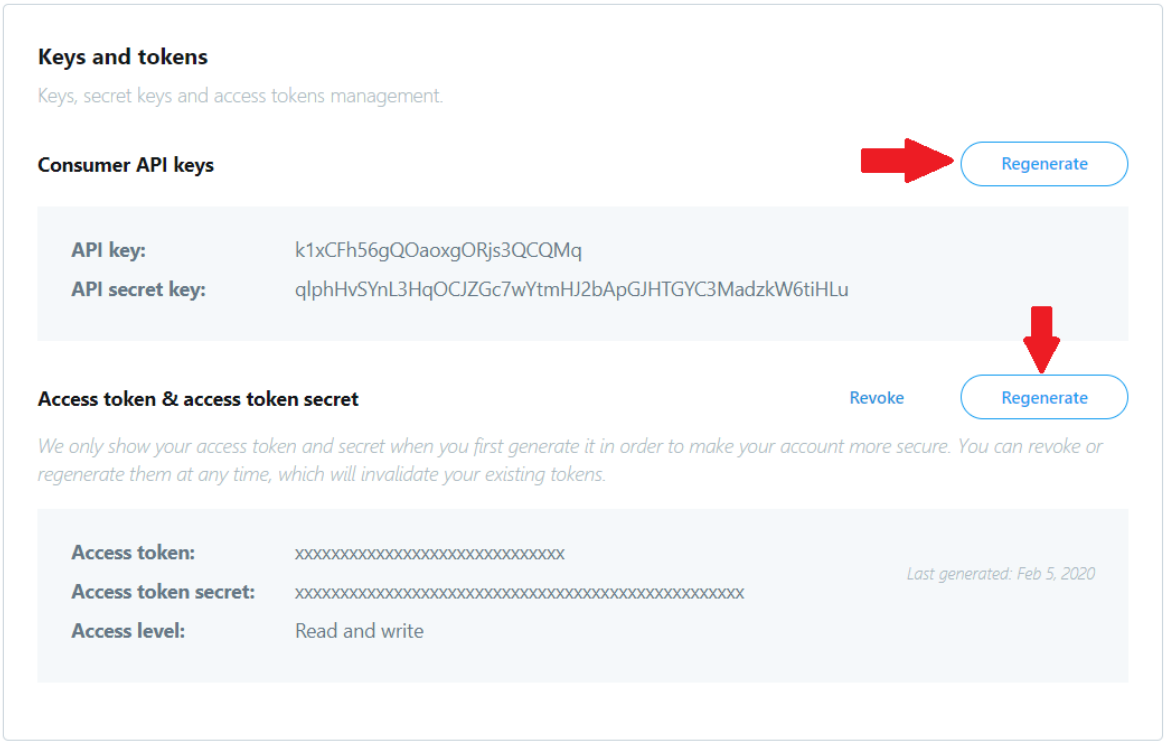

Imagen 3 – Generar keys and tokens

Manejo de Tweets: Apache Kafka - Apache Zookeeper

Como parte de la arquitectura, es importante tener un orquestador de los mensajes que se reciben en un lapso de tiempo específico antes de ser procesados, la cual es la responsabilidad de Apache Kafka, sin embargo este sistemas de manera distribuida (maestro - esclavo) para lo cual usa Apache Zookeeper como el maestro encargado de la orquestación de muchos nodos kafka. Podemos encontrar definiciones muy exactas de cómo nacieron ambas soluciones:

"Apache Kafka es un proyecto de intermediación de mensajes de código abierto desarrollado por LinkedIn y donado a la Apache Software Foundation escrito en Java y Scala. El proyecto tiene como objetivo proporcionar una plataforma unificada, de alto rendimiento y de baja latencia para la manipulación en tiempo real de fuentes de datos. Puede verse como una cola

de mensajes, bajo el patrón publicación-suscripción, masivamente escalable concebida como un registro de transacciones distribuidas, lo que la vuelve atractiva para las infraestructuras de aplicaciones empresariales."

"Apache ZooKeeper es un proyecto de software libre de la Apache Software Foundation, que ofrece un servicio para la coordinación de procesos distribuido y altamente confiable que da soluciones a varios problemas de coordinación para grandes sistemas distribuidos. ZooKeeper es un subproyecto de Hadoop. La arquitectura de ZooKeeper soporta alta disponibilidad a través de servicios redundantes. Los clientes pueden así preguntar a otro maestro ZooKeeper si el primero falla al responder. Los nodos ZooKeeper guardan sus datos en un espacio de nombres jerárquico, como hace un sistema de archivos o una dato estructura (estructura de datos) trie. Los clientes pueden leer y escribir desde/a los nodos y de esta forma tienen un servicio de configuración compartido."

A fin de comprender cómo utilizar ambas soluciones vamos a desarrollar una serie de paso a paso para poder utilizarlas a nivel local:

1. Instalar Apache Kafka con Zookeeper:
    a. Ir a https://kafka.apache.org/downloads
    b. Descomprimir el Archivo .tgz
    c. Colocar la carpeta llamada "kafka_2.11-2.3.0" en el directorio C:/

2. Inicializar Zookeeper
    a. Ir a la carpeta de Kafka "cd C:\kafka_2.11-2.3.0"
    b. Inicializar con el comando
        "bin\windows\zookeeper-server-start.bat config\zookeeper.properties"

3. Inicializar Kafka
    a. Ir a la carpeta de Kafka "cd C:\kafka_2.11-2.3.0"
    b. Inicializar con el comando
        "bin\windows\kafka-server-start.bat config/server.properties"

4. Con el maestro y al menos un nodo inicializado es hora de crear la canalización por la cual los mensajes van a ir llegando, a esto comúnmente se le llama "topic":

    "bin\windows\kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic twitter_stream"

5. Para verificar que el canal fue creado con éxito, es posible ver la lista de canales:

   “bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092”

6. Para verificar que hay mensajes llegando a kafka es posible consumir los mensajes mediante el siguiente comando:

   “bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic twitter_stream --from-beginning”

7. Y solo por si acaso se desea eliminar la canalización es posible mediante:

   “bin\windows\kafka-topics.bat --delete --bootstrap-server localhost:9092 --topic twitter_stream”

   Para más información, favor ver el video adjunto a la entrega de este trabajo, disponible en el siguiente link:

Proceso y Análisis de Tweets: Spark Streaming

Spark Streaming es una de los componentes que son parte de marco de trabajo Apache Spark, es utilizado para el consumo y análisis de información en tiempo real.

"Apache Spark es un framework de computación en clúster open-source. Fue desarrollada originariamente en la Universidad de California, en el AMPLab de Berkeley. El código base del proyecto Spark fue donado más tarde a la Apache Software Foundation que se encarga de su mantenimiento desde entonces. Spark proporciona una interfaz para la programación de clusters completos con Paralelismo de Datos implícito y tolerancia a fallos.

*Apache Spark se puede considerar un sistema de computación en clúster de propósito general y orientado a la velocidad. Proporciona APIs en Java, Scala, Python y R. También proporciona un motor optimizado que soporta la ejecución de grafos en general. También soporta un conjunto extenso y rico de herramientas de alto nivel entre las que se incluyen Spark SQL (para el procesamiento de datos estructurados basada en SQL), MLlib para implementar machine learning, GraphX para el procesamiento de grafos y Spark Streamin*g"

A nivel de código, la solución planteada utiliza tres diferentes etapas que son esenciales para su creación:

1. **Autenticación:** En esta etapa se desarrolló el código necesario para establecer una comunicación exitosa entre la aplicación y el API de twitter con el fin de recibir los mensajes que se postean en dicha aplicación.

2. **Productor:** En esta etapa se desarrolló el código necesario para recibir los mensajes del API de twitter y ponerlo en una cola de kafka.

3. **Consumidor:** En esta etapa se desarrolló el código necesario para consumir los mensajes en kafka, procesarlos, analizarlos y dejar los resultados en un archivo en HDFS.

Almacenamiento: HDFS

Cuando se habla de procesamiento en tiempo real, se habla de grandes volúmenes de información que son generados minuto a minuto, sin embargo no toda la información que se recibe es crucial para el análisis que se desea realizar , es por ello que a la hora de almacenar la información hay que buscar almacenar aquello que realmente importante. Como parte de esta solución los resultados serán guardados en HDFS en formato parquet, para que cuando se deseen hacer análisis posteriores se puedan realizar sin ningún problema. Por cierto cuando hablamos de HDFS, hablamos de:

"HDFS es el sistema de ficheros distribuido de Hadoop. El calificativo distribuido expresa la característica más significativa de este sistema de ficheros, la cual es su capacidad para almacenar los archivos en un clúster de varias máquinas.

Esta característica es imperante cuando se pretenden almacenar grandes cantidades de datos, puesto que en general no es posible almacenar cientos de terabytes o petabytes en una única máquina."

Conclusiones

Durante el desarrollo de este proyecto se han obtenido las siguientes conclusiones:

- En este trabajo fue posible investigar e integrar distintas aplicaciones de Big Data en una sola solución, permitiendo la interconexión de cada uno de ellos y así permitiendo el objetivo principal que era el análisis y el procesamiento de información a gran escala y en tiempo real.

- Realizar la parte más sencilla del proyecto tomo más tiempo del esperado, dado que las políticas que mantiene Twitter para los permisos de desarrollo son estrictas.

Referencias

Apache Kafka, Extraído de: https://es.wikipedia.org/wiki/Apache_Kafka

Apache ZooKeeper, Extraído de: https://es.wikipedia.org/wiki/Apache_ZooKeeper

Apache Spark, Extraído de: https://es.wikipedia.org/wiki/Apache_Spark

Hadoop Distributed File Syste, Extraído de:
https://es.wikipedia.org/wiki/Hadoop_Distributed_File_System