.

Tecnologías de Procesamiento

Big Data

ÍNDICE

INTRODUCCIÓN

Nuestro proyecto, enfocado en optimizar la estrategia de Trading de Broskis Brokers dentro del sector de Tecnologías de la Información del SP-500, se encuentra en pleno desarrollo bajo la metodología ágil de Scrum. Este tercer sprint marca un momento crucial en nuestro proceso, donde nos enfocamos en objetivos específicos para la integración y análisis de datos en tiempo real, así como en la implementación de herramientas nuevas como MySql para nuestro flujo de trabajo.

SPRINT 2 RETROSPECTIVE

Después de concluir el segundo sprint, nos reunimos para evaluar nuestro desempeño y los resultados obtenidos. Durante esta sesión, analizamos tanto los aciertos como los errores cometidos, así como las principales dificultades que enfrentamos. El propósito de esta reunión fue claro: aprender de los errores cometidos, de manera que aprender de ellos nos permita mejorar en futuros sprints.

En primer lugar, es gratificante destacar que logramos finalizar todas las tareas dentro del plazo establecido, cumpliendo así con los objetivos previamente definidos. La coordinación entre los miembros del equipo fue excelente, y la distribución de tareas se realizó de manera equitativa, evitando que alguien se viera sobrecargado en cuanto a horas dedicadas al proyecto.

Sin embargo, identificamos un punto negativo que merece ser mencionado: nos vimos apurados de tiempo al intentar implementar una idea compleja. Esta idea consistía en la implementación de un protocolo, que, aunque interesante, no estaba dentro de los requerimientos específicos del sprint.

En resumen, el segundo sprint fue un éxito en términos de cumplimiento de objetivos y trabajo en equipo. Aprendimos valiosas lecciones que nos ayudarán a optimizar nuestro desempeño en futuros proyectos, como la importancia de mantener un equilibrio entre la complejidad de las ideas y los requisitos establecidos, así como la necesidad de una comunicación más frecuente y transparente dentro del equipo.

SPRINT 3 PLANNING

Para este sprint, nos enfocaremos en consolidar y mejorar nuestra infraestructura de datos para optimizar nuestra estrategia de Trading. Nuestro objetivo principal será completar la creación de la base de datos en MySQL, establecer una conexión robusta y un flujo de trabajo eficiente en Apache NiFi, realizar ajustes en el formato de los datos y asegurar la subida efectiva de datos a Elasticsearch.

Comenzaremos implementando la creación de la base de datos en MySQL, incorporando los filtros necesarios para manejar tanto los datos históricos como los datos en tiempo real. Esta tarea es crucial ya que proporcionará la base fundamental para la gestión y análisis de los datos en nuestro sistema.

Posteriormente, nos centraremos en establecer la conexión entre Apache NiFi y Kafka, garantizando una transferencia de datos fluida y confiable. Configuraremos el flujo de trabajo en NiFi para optimizar la transformación y carga de datos en tiempo real, asegurando que el proceso sea eficiente y escalable.

A continuación, nos ocuparemos de realizar los cambios necesarios en el formato de los datos para cumplir con los requisitos de Elasticsearch y con el cambio de formato propuesto por los stakeholders, para que sea más simple de cara a su uso como inputs para el entrenamiento de los modelos.

Finalmente, configuraremos el conector desde Apache NiFi a Elasticsearch para garantizar la persistencia y accesibilidad de los datos analizados. Verificaremos la efectividad de la subida de datos y nos aseguraremos de que estén disponibles para su consulta y análisis en todo momento, fortaleciendo así nuestra estrategia de Trading con una infraestructura de datos sólida y eficiente.

ORGANIZACIÓN DE TAREAS

Debido al éxito de este segundo sprint no hemos valorado la opción de un cambio de roles, por ello se encargaron de la designación de tareas para este segundo sprint. Quedando el esquema de la siguiente forma.

TO DO	RESPONSIBLE	DONE
MySql-Hive	Antonio Mora	✔
Nifi-Elastic Search	Mario Kroll y Nicolas Villagrán	✔
Documentación	Pablo Díaz	✔

TABLA 1

CREACIÓN DE LA DB

En primer lugar, hicimos una primera pasada por la tabla de **company_500** para ver las columnas que serán importantes y ver sobre qué columnas debemos filtrar:

```
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| CIK              | varchar(10)  | YES  |     | NULL    |       |
| Company          | varchar(100) | YES  |     | NULL    |       |
| Symbol           | varchar(20)  | YES  |     | NULL    |       |
| GICSSector       | varchar(100) | YES  |     | NULL    |       |
| GICSSubIndustry  | varchar(100) | YES  |     | NULL    |       |
| Location         | varchar(100) | YES  |     | NULL    |       |
| Dateadded        | date         | YES  |     | NULL    |       |
| Founded          | varchar(40)  | YES  |     | NULL    |       |
| Portfolio        | varchar(20)  | YES  |     | NULL    |       |
| Price            | float        | YES  |     | NULL    |       |
| chg              | varchar(20)  | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
```

CAPTURA 1

De esta tabla nos interesa únicamente los datos de las compañías que pertenezcan a nuestro sector, es decir, al de "Information Technology". Por ello, realizaremos una filtración en la tabla **company_500**. Todo esto se puede hacer de una vez empleando el siguiente comando:

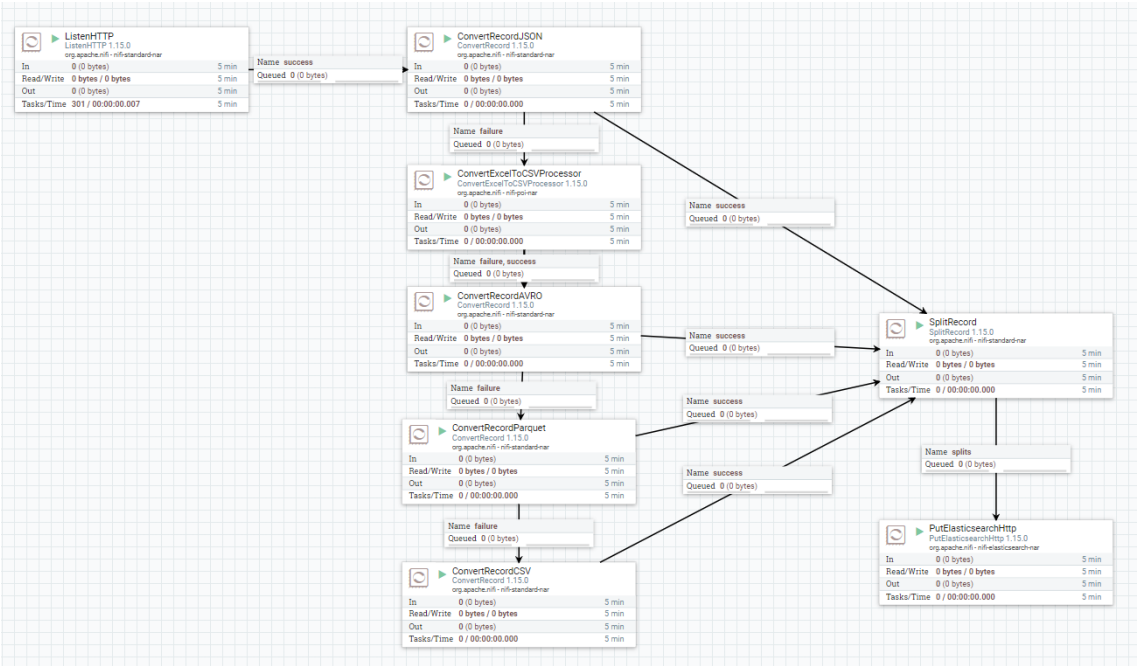
```
sqoop import -m 1 --table company_500 --connect
jdbc:mysql://manager01:3307/mydatabase?useSSL=no --username myuser --password userpass --
hive-import --hive-database 'technology_information' --hive-table 'technology_information_data' --
where "GICSSector = 'Information Technology'"
```

Este comando tiene varias partes. En primer lugar, nos conectamos a la base de datos donde se encuentra actualmente la tabla **company_500** (llamada *mydatabase*). Especificando el usuario y contraseña para poder tener acceso a ella, sqoop tiene comandos implementados que nos permiten hacer la carga de la tabla a nuestra base de datos en hive de manera directa. La banderilla –hive-database nos permite identificar la base de datos sobre la que trabajaremos (importante que ya tiene que existir) y –hive-table nos permite especificar el nombre de la tabla sobre la que queremos cargar los datos (si no existe, la crea). Por último, la banderilla –where nos permite realizar una filtración en la tabla antes de cargarla, dejándonos especificar que solo queremos las compañías del sector "Information Technology" en este caso.

CAMBIO EN EL FORMATO DE DATOS

Los archivos en formato JSON, si los guardamos en un dataframe, la fecha se guarda por defecto en formato de milisegundos. Sin embargo, ElasticSearch solo permite datos de fecha en el formato ISO (YYYY-MM-DD). Por esto, hemos tenido que ser más precavidos a la hora de guardar los datos y hemos tenido que hacer ciertas modificaciones en los ficheros de Python existentes.

ESTRUCTURA NIFI Y CARGA DE DATOS A ELASTICSEARCH



CAPTURA 2

El siguiente flujo de trabajo, lo necesitaremos para completar la Parte 3 del proyecto, que consiste en subir y almacenar los datos en ElasticSearch. Estos datos, sin embargo, se pueden enviar en formato avro, parquet, csv, json, orc y xlsx. Esto obliga a hacer uso de la herramienta Apache Nifi y montar una estructura de procesadores como se puede ver en la Captura 2. El funcionamiento de este es el siguiente:

El flujo de trabajo presentado en Apache NiFi se ha diseñado para optimizar la ingesta y almacenamiento de datos en Elasticsearch, manejando una variedad de formatos como Avro, Parquet, CSV, JSON y XLSX. La operación comienza con un procesador HTTP Listen, que recibe archivos mediante peticiones HTTP POST, utilizando el comando "`curl -X POST -F "file=filename.filetype" worker01:18114/contentListener`" worker01:18114/contentListener para la transmisión desde una ubicación local.

Una vez recibido el archivo, NiFi ejecuta una serie de validaciones para determinar su formato. Si el archivo es de tipo JSON, se procede a una división de este (SplitJson) para facilitar la carga incremental en Elasticsearch, evitando la sobrecarga del sistema. En caso de un fallo en la detección del formato JSON, el flujo redirige el archivo hacia el siguiente procesador encargado de tratar con archivos XLSX, convirtiéndolos a CSV. Si el archivo original es de formato Avro y falla en el procesador de XLSX, continúa su camino hacia el procesador específico para Avro y así sucesivamente hasta que se procesa correctamente según su tipo.

Cada procesador está configurado para transformar los datos a formato JSON tras un resultado exitoso, dado que Elasticsearch requiere que los datos se carguen en este formato. En el caso de que se encuentre un archivo CSV, este se procesa directamente y, en la inusual circunstancia de un error, se elimina del flujo.

Este enfoque modular garantiza la flexibilidad y la robustez del sistema, permitiendo la expansión o modificación del flujo para adaptarse a nuevos formatos de datos o requerimientos futuros sin interrumpir la operativa existente. Este proceso es válido para archivos de tipo avro, JSON, CSV, parquet y XLSX. Sin embargo, con archivos de tipo orc, la tarea se complica un poco más. En este caso, hemos optado por subir directamente el archivo orc a Hive, sin tener que pasar por ElasticSearch. Esto se debe a una simple razón: trabajar con ficheros orc en Nifi es más tedioso que con cualquiera de los otros. Para llevar a cabo dicha tarea, hemos tenido que crear una tabla en Hive dentro de nuestra base de datos que tuviera un esquema como el de nuestro fichero orc. Para crear la tabla, hemos hecho uso del siguiente comando:

```
hive> CREATE TABLE IF NOT EXISTS data_orc_5(Fecha string, CIK int, Symbol string, Open double, High double, L
ow double, Close double, Volume double, Dividends double, StockSplits double) ROW FORMAT DELIMITED FIELDS TER
MINATED BY ',' STORED AS ORC;
OK
```

CAPTURA 3

Por último, para importar los datos a la tabla, hemos hecho uso del siguiente comando:

```
hive> LOAD DATA INPATH '/tmp/information_technology/data_5.orc' INTO TABLE data_orc_5;
Loading data to table technology_information.data_orc_5
Table technology_information.data_orc_5 stats: [numFiles=1, totalSize=429193]
OK
Time taken: 0.374 seconds
```

CAPTURA 4

PRÓXIMOS OBJETIVOS

Nuestros próximos objetivos, vendrán de la mano del desarrollo de modelos predictivos y muy probablemente del análisis de datos con interfaces como la de Kibana para poder realizar informes más precisos y analíticos, con el objetivo de que los stakeholders tengan una visión más simplificada de los datos.