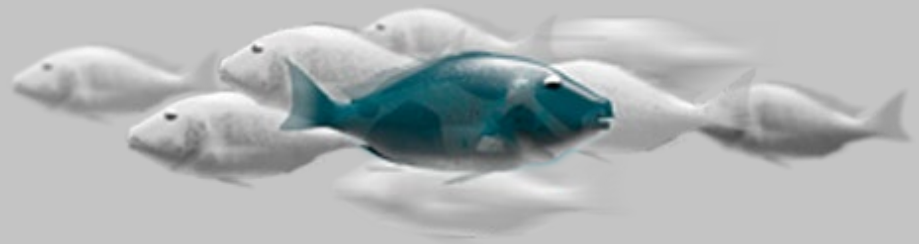
EDEM



# Máster en Data Analíticos

Apache Spark Sessions 2023-24

ETL Use Case

Pablo Pons Roger

# ETL Use Case

# Pasos

El propósito de este ejercicio es que cada uno de vosotros desarrolle el código Spark para una aplicación que simule un caso de uso real, una ETL como las que podemos encontrar en cualquier proyecto del día a día.

Para ello, debemos seguir los siguientes pasos:

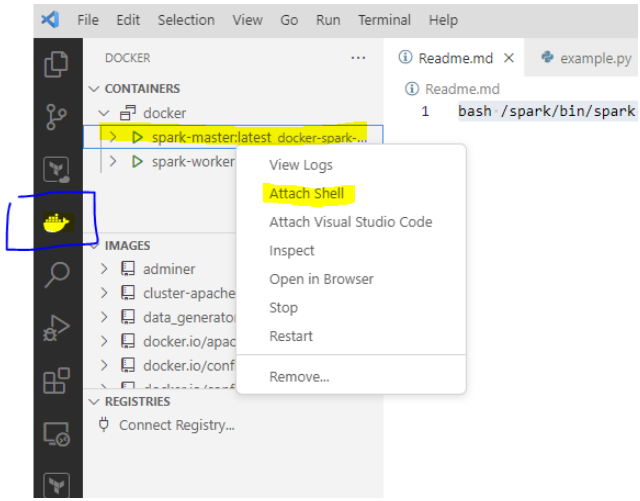
1. Entrar en la carpeta de e2e_spark/. Es recommendable abrirla con VS Code para facilitar los siguientes pasos.
2. Abrir una consola, entrar en la carpeta ./docker, y ejecutar:

   ```
   $ docker-compose up –d
   ```

   Importante: previamente temenos que iniciar el Docker Desktop (iniciar el daemon de Docker).

# Pasos

3. Una vez hecho esto, entramos en el Docker Desktop y comprobamos que se han creado correctamente dos contenedores: spark-master-1 y spark-worker-1
4. Ahora que ya temenos los contenedores creados, vamos a ejecutar un app Spark de prueba. Para ello debemos abrir una terminal en el contenedor spark-master-1 que acabamos de crear. Si tenemos instalada la extensión de Docker para VS Code es tan fácil como abrirla, y encima del contenedor hacer click derecho y Attach Shell.



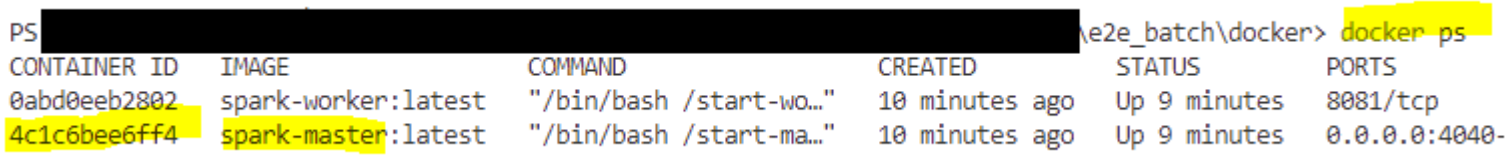
# Pasos

Si no, siempre podemos lanzar en la consola que teníamos abierta:

$ docker exec –it <id_container> sh

Donde el <id_container> es el identificador del contenedor spark-master-1. Podemos ver este id en el Docker Desktop o bien ejecutando en consola

$ docker ps

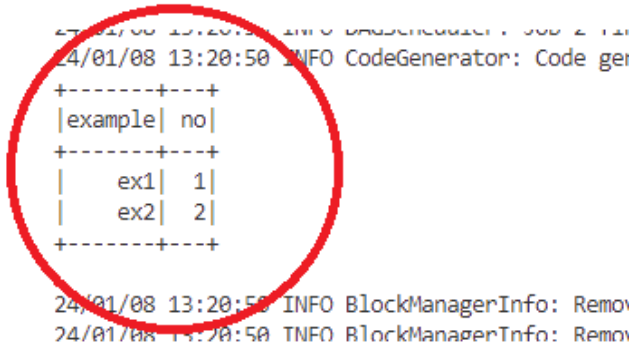


# Pasos

5. Ahora ya podemos ejecutar la app Spark de prueba. Para ello, lanzamos en la terminal del contenedor:

   $ bash /spark/bin/spark-submit --master local opt/project/src/python/example.py

   Y deberemos ver en el log el siguiente output:



# Pasos

6. Una vez hecho esto ya nos queda elegir nuestra fuente de datos, diseñar nuestra lógica, y desarrollarla. En este artículo teneis un listado de algunas fuentes de datos públicas que pueden ser útiles: https://www.dropbase.io/post/top-11-open-and-public-data-sources
7. Los resultados del análisis los escribiremos en la propia carpeta de resources como otro CSV.
8. Para los que vayan más avanzados, proponemos ampliar el ejercicio incluyendo una BBDD SQL, por ejemplo MySQL, y escribiendo en ella en lugar de escribir en el propio contenerdor.

**IMPORTANTE**, la lógica desarrollada debe tener al menos las siguientes transformaciones: **Filter, WithColumn, Join GroupBy**.

# Pasos

Recordad que para ejecutar vuestro propio código, debereis:

- Introducir los archivos (sources) en la carpeta resources y leer en ese path
- Modificar el comando y donde pone example poner el nombre de vuestro .py

```
$ bash /spark/bin/spark-submit --master local opt/project/src/python/example.py
```

Como tenemos que llevar a cabo joins, es probable que necesitemos dos sources. En ocasiones es complicado encontrar buenas fuentes de datos, de manera que para facilitar la tarea, solo si es necesario, podeis “mokear” una fuente de datos. Por ejemplo, tengo datos de transacciones y quiero hacer un cruce con diferentes divisas. Para ello, puede resultar útil la siguiente herramienta: https://www.mockaroo.com/. Al menos uno de los sources debe ser real (no mockeado).

# Entregable

El entregable será por tanto:

- El archivo .py con las transformaciones
- Los archivos (CSV, JSON) que utilicemos como Fuente
- Un pequeño readme o un txt con una breve explicación del use case, y de la lógica. Muy breve, un párrafo aprox
- Los que hayan realizado el ejercicio ampliado también el docker-compose.yml modificado
- Se agradecerán también comentarios sobre lo que se hace en el propio .py

Eso es todo, suerte ☺