Trabajo Práctico Final

El siguiente trabajo práctico se realizará de manera individual.

El trabajo evaluará los conocimientos adquiridos en el Módulo Data Applications.

Las tecnologías a utilizar son **RDS, S3, Airflow (opcionalmente MWAA) y Python 3.6+.**

Se debe enviar un email con la entrega a jpamplie@itba.edu.ar, jgoldman@itba.edu.ar y plorenzatto@itba.edu.ar antes de la fecha estipulada.
El email con la entrega debe tener en el subject **[TP ML Engineering]**

La fecha de entrega es el martes 01 de Marzo de 2022. La fecha límite de entrega tardía es el lunes 03 de Abril de 2022 a las 00hs.

Contexto:
Acaba de ser contratado como el primer ingeniero de datos de una pequeña empresa de viajes.
Su primera tarea para usted fue demostrar el valor y los conocimientos que se pueden generar a partir de las canalizaciones de datos.

Su plan es que una vez que demuestre lo valiosos que pueden ser los datos, comenzarán a invertir en el uso de un proveedor de instancias en la nube.
Por ahora, su propia computadora tendrá que hacerlo.

Objetivo:
Crear un DAG de Airflow que actúe de ETL para extraer extraiga datos estáticos S3 y los cargue en una base de datos de Postgres.

Datos a utilizar:
Para llevar a cabo el desarrollo se utilizará el dataset de demoras y cancelaciones de viajes aéreos de Kaggle que será hosteado en un bucket creado por el alumno en S3.

Lo primero será obtener los datos siguiendo estos pasos:

1. Instalar el cliente de Kaggle: `pip install kaggle`

2. Instalar el cliente de aws siguiendo estas instrucciones acorde a su sistema operativo.
3. Configurar las credenciales siguiendo estas instrucciones
4. Bajar datos de Kaggle:

```
# cd to your local directory
cd /path/to/dataset/
$ mkdir -p minio/data/flights-bucket
# Download zipped dataset from kaggle
$ kaggle datasets download -d
yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018
# Unzip files
$ unzip airline-delay-and-cancellation-data-2009-2018.zip -d raw/
# Remove zipped data to save space
$ aws s3 sync raw/ s3://[BUCKET_ALUMNO]/tp/
# Remove zipped data to save space [optional]
$ rm airline-delay-and-cancellation-data-2009-2018.zip
```

En este punto al correr el comando el siguiente comando debería aparecer un archivo CSV por año en el directorio de s3:
`aws s3 sync raw/ s3://[BUCKET_ALUMNO]/tp/`

Desarrollo:

1. Configurar Airflow para que corra en AWS.
   Esto se puede hacer de varias maneras, puede setearse en una instancia EC2 o bien utilizar el servicio de Airflow manejado por AWS MWAA.

2. Crear una instancia RDS de Postgres. Esta instancia será utilizada como DB en los puntos siguientes.

3. Desarrollar un DAG de Airflow con schedule anual que:
   - Calcule el promedio del tiempo de demora de salida (columna `DEP_DELAY`) por aeropuerto de salida (columna `ORIGIN`) y dia

   - Utilice un algoritmo de detección de anomalías para identificar por cada aeropuerto si hubo algún día con demoras fuera de lo normal.
     Para esto se puede utilizar scikit-learn y correrlo mediante un PythonOperator o bien utilizar SageMaker.

   - Utilizando los datos del punto anterior por cada aeropuerto produzca un gráfico desde Python usando Pandas o Matplotlib en el cual se

pueda ver la cantidad de vuelos de cada día con alguna indicación en los días que fueron considerados anómalos.

- Hecho lo anterior se debe cargar la data sumarizada junto con un indicador para la fila correspondiente de cada día para indicar si para ese día en un aeropuerto particular las demoras estuvieron fuera de lo esperable. Asimismo los gráficos generados anteriormente deben ser almacenados en S3 en un path fácilmente identificable por año y aeropuerto analizado.

4. Desarrollar una visualización de los datos cargados. Esto se puede hacer alternativamente de dos maneras:
   - Configurar Superset para que se levante utilizando Docker y muestre un dashboard. En caso de utilizar Docker o Docker Compose es necesario incluir instrucciones y archivos necesarios para llevar a cabo la configuración.
   - Configurar un dashboard con el servicio AWS Quicksight. En este caso es necesario incluir en la documentación del proyecto los pasos requeridos para que el servicio quede operativo.

Notas:

- El DAG debe funcionar para cualquiera de los años 2009 a 2018 incluidos en el dataset. Tener en cuenta que si se corre dos veces para el mismo año podría haber una duplicación de datos. Cómo podríamos resolverlo?

Entregables:

Repositorio de código Git que contenga:

- Archivo README.md con:
  - Descripción de:
    - Descripción del problema que se quiere resolver.
    - Las partes que incluye la arquitectura desarrollada.
  - Instrucciones para crear y configurar los recursos necesarios para reproducir el desarrollo y dejarlo operativo.
- Imagen con el diagrama de arquitectura de la solución armada en AWS donde se vean claramente:
  - Los servicios utilizados
  - Las fuentes y destinos de datos
  - Los flujos entre fuentes de datos, servicios y destinos finales.
- Scripts/archivos de Bash y Python relevantes.