ITBA Escuela de INNOVACIÓN

EVOLUCIÓN CONTINUA

DIPLOMATURA EN CLOUD DATA ENGINEERING

Consigna Trabajo Práctico Integrador - PDA

www.itba.edu.ar

¿Qué vas a lograr con este TP?

1. **Afianzar** todos los **conceptos** vistos en este curso.
2. Aplicar los distintos **conceptos teóricos y prácticos** vistos durante el módulo para llevar a la **práctica un producto final**.
3. Sumar una **herramienta** de presentación a su **portfolio** personal.
4. Terminar de **despejar dudas** durante la práctica, mediante el **acompañamiento** del staff educativo del curso.
5. **Obtener experiencia** en un mini proyecto que podría existir en el día a día de **cualquier Data Engineer en cualquier compañía**.

¿Qué vas a aplicar en este TP?

1. **Código Python** pensando en estándares de calidad y aplicando buenas prácticas.
2. **Testing unitarios**.
3. Creado de un **repo en Github**, usar **commits** y **PRs** para avanzar en el proyecto y dejar un historial.
4. CI/CD a través de **Github Actions**.
5. Lanzado de entornos a través de **Docker**.
6. Utilizar **Airflow** para correr un script de **batch processing**.
7. Utilizar el módulo *requests* para consultar **APIs REST** y extraer datos. O en su defecto un SDK que permita al alumno obtener los datos que desee.
8. Utilizar el módulo *pandas (o similar)*, o algún módulo similar para **leer y transformar datos**.
9. Utilizar el formato *parquet* para guardar datos semi estructurados para generar un área de *staging* en el Data Warehouse.
10. Utilizar **AWS Redshift** como Data Warehouse para almacenar el resultante del **ETL**.

Descripción del TP

El alumno deberá crear un script de **Python** para extraer datos de una **API** pública (o privada, proporcionando las credenciales). Este script deberá correr en un DAG de Airflow (v2.x). Airflow deberá ser deployado/lanzado mediante **Docker** (usando un archivo Dockerfile o un docker-compose).

El trabajo deberá existir en un repositorio de **Github** (público o privado). El repositorio deberá tener mínimo 1 **Pull Request** creada y *mergeada*, que demuestre la aplicación de **tests automáticos en el código** correspondiente a ese PR.

Estos tests serán **unitarios**, y testearán algunos de los métodos creados por el alumno para procesar los datos. El alumno puede utilizar el módulo ***que considere adecuado*** para correrlos.

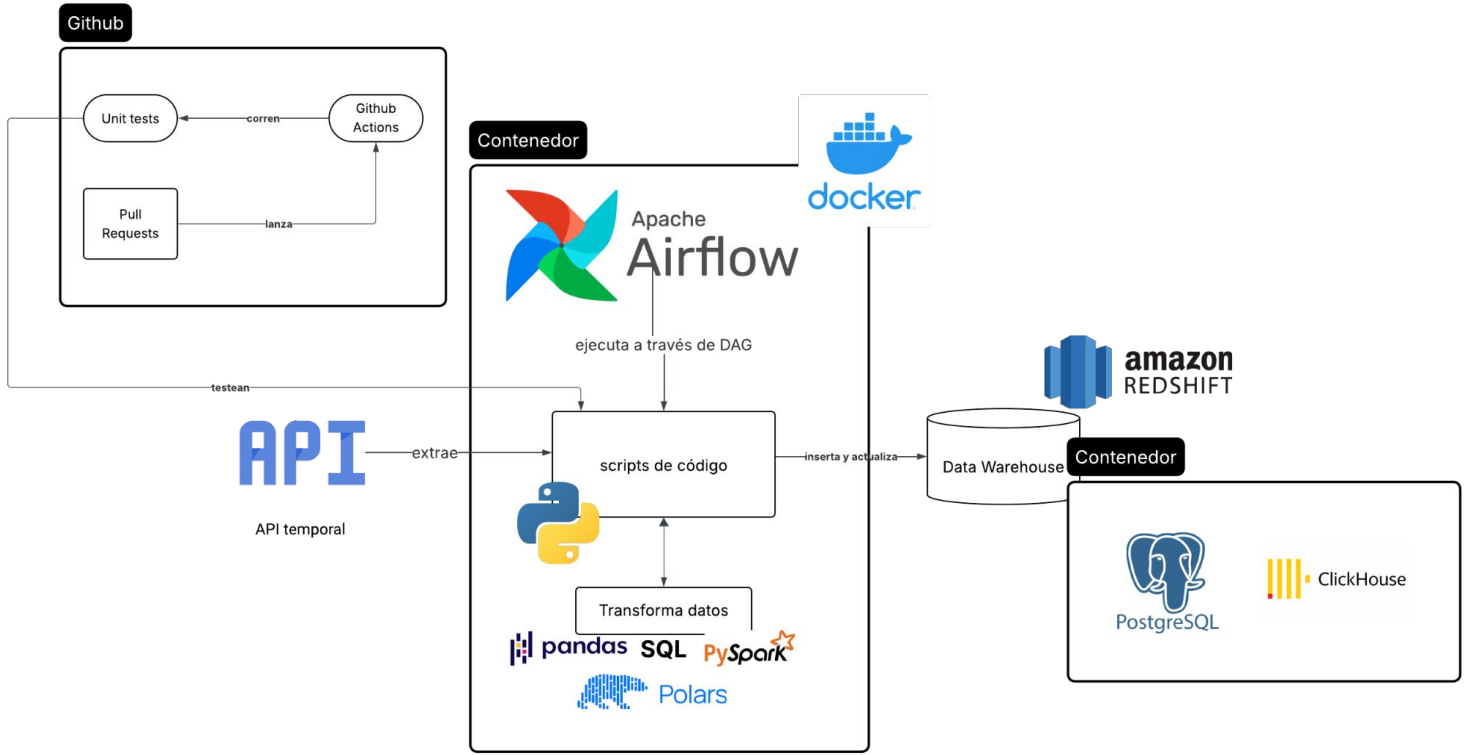
Con respecto al pipeline, el mismo deberá extraer datos utilizando requests a una **API** (o usando un módulo/SDK que le permita obtener estos datos de igual forma). Los datos deberán ser cargados utilizando alguna herramienta como **pandas o similar (polars, dask, etc)**.

Los datos deberán ser **transformados, limpiando, agregando o modificando datos** de la forma que el alumno desee. Se valorará el hecho de que hayan **distintos tipos de transformaciones**. Por ejemplo, se podrá obtener, de un dataset inicial, agregar columnas al mismo, modificar datatypes de las columnas; eliminar columnas; rellenar valores nulos; aplicar mapeos; **mergear con otro dataset**; combinar datos con otra API, un archivo local, o una tabla del cluster de Redshift; entre otras transformaciones.
Se recurre a la creatividad del alumno y la adaptabilidad de los datos que eligió utilizar.

Los datos resultantes del script deberán ser cargados en un esquema dentro de una base de datos tipo OLAP **(Nosotros proveemos Redshift)**. No es necesario que la totalidad de las transformaciones se realicen dentro del pipeline de Python, sino que se podrá utilizar SQL que se lance dentro de Airflow, para realizar la transformaciones dentro de la base de datos.

ITBA

Arquitectura básica del TP



Requisitos mínimos

Deberán estar presentes en el repositorio de GitHub:

- Un archivo **README.md** en el directorio base del repositorio. El mismo deberá **enumerar los pasos que permitan probar el TP** y deberá explicar las características del trabajo realizado. Se deberán especificar, al menos en alto nivel, los archivos que componen el mismo y su funcionalidad, las credenciales necesarias (para la API, por ejemplo, y enviárlas por privado mail o Slack), y cualquier otro paso para hacer funcionar al TP.
- Un **Dockerfile o un docker-compose** para lanzar el entorno desarrollado.
- Un archivo **Python con el DAG escrito**. Pueden estar presentes **tantos archivos como sean necesarios** para lograr un código modular y ordenado.
- Consultar una **API variable en el tiempo**.
- Al menos un archivo con el workflow de testing de GitHub Actions en el directorio **.github/workflows** del repositorio.
- Una carpeta **tests** que posea 2 tests unitarios mínimo, testeando funciones o rutinas que se encuentren en el TP. Estos se deberán ejecutar con GitHub Actions, y se deberán poder ejecutar de forma local también (Estos tests deberán testear el resultado de una transformación que ocurra dentro de una función, como mínimo deben tener un *assert* en cada uno de los tests).
- Un Pull Request (mínimo, se recomiendan más) en **estado "merged"** que muestre los **tests corridos y aprobados** (es decir, que hayan pasado). La pull request **puede ser de cualquier feature o cambio**, puede ser un cambio en la transformación de los datos, nuevo código para pasar del dataframe a un archivo parquet, o código para subir el archivo a la nube. Lo importante es que el alumno **simule un PR bien hecho**, con la descripción del cambio y mostrando los tests corriendo.
- Esqueleto de archivo de entorno **.env/.cfg/.yaml** para **nosotros llenar con nuestras credenciales** y correr el código.
- En general, se espera que el código **sea fácilmente legible**. En caso de precisar comentarios extra para explicar un procedimiento, los mismos son bienvenidos.
- El hecho de encontrar credenciales expuestas dentro del repositorio es justificativo para la no aprobación del TP.
- **La suma de 5 puntos en requisitos extra**.

Requisitos Extra (sumá 5 puntos min.)

Las siguientes son **algunas** de las características que le agregarán más valor al TP, harán un proyecto más comprensivo y nutrido y serán tenidas en cuenta al momento de la corrección:

- Que se consulten más de una API para extraer datos, y que se consoliden o combinen datos, o para generar un modelo más complejo en Redshift (**3 puntos)**.
- Crear una base de datos SQL (por ejemplo, PostgreSQL), a la cual se le puede cargar una tabla manualmente, y utilizarla en el script.(Se puede colocar la creación en el docker-compose o dejando instrucciones de cómo crearla) (**3 puntos**).
- Utilización de SQL en Redshift para la creación y transformación de tablas. Por ejemplo, generando otras tablas o vistas al estilo Data Marts (**2 puntos**).
- Utilización de decorators en los scripts de Python (**1 punto**).
- Aplicación de *typing* en las funciones generadas, con instrucciones para hacer un chequeo de esto como se vió en clase (**1 punto**).
- Realización de un esquema estrella (**3 puntos**).
- Realización de una tabla SCD2 (**5 puntos**).
- Implementación de una arquitectura estilo ELT, donde se haga una primer carga de los datos sin transformar (o con una transformación mínima para poder realizar la carga), y después se transformen hacia otra tabla o schema (**3 puntos**).

Requisitos Extra (sumá 5 puntos min.)

- Un archivo Makefile que facilite al usuario a correr el TP. Por ejemplo, especificando un comando "***make run-project***" para ejecutar un "***docker build***" y "***docker run***" o directamente un "***docker-compose up -d*** …". Este archivo también puede tener comandos para realizar otras acciones para probar el TP (**1 punto**).
- Múltiples Pull Request con descripciones de cambios apropiadas y con tests unitarios (**2 puntos**).
- Tests end to end (**3 puntos**).
- Una Github Actions que realice linting (**1 punto**).
- Generación de docstrings y con instrucciones para chequear que están completos como se vió en clase (**2 puntos**).
- Utilización de Airflow Variables que modifiquen el proceso de ETL (**2 puntos**).
- Algún framework dado o no en clase que aporte un valor significativo al proyecto (Great Expectations, dbt, etc.) (**5 puntos**).
- Sistema de notificación (de alertas o estado del ETL por ejemplo) que se comunique mediante mail o por Slack (**3 puntos**).

SUPER IMPORTANTE

ELEGIR UNA API QUE SEA TEMPORAL SÍ O SÍ. NO ES JODA. Hemos reprobado gente por esto.

Ejemplos de APIs temporales:

- Stocks, o instrumentos financieros que cambian su valor en el tiempo.
- Clima, humedad, valores climáticos en el tiempo.
- Precios de hoteles en alguna ciudad o país. O de vuelos.
- Datos biométricos o demográficos. Contagios de covid, viruela de mono, etc.
- Datos tipo IOT: sensores de medición de alguna cosa.
- Cantidad de publicaciones, seguidores, likes, comentarios de algún usuario/s en alguna red social.

Ejemplos de APIs que NO VAN:

- Superhéroes de Marvel / personajes de star wars / etc.
- Equipos de fútbol de alguna liga en particular. O jugadores de ese equipo.
- Razas de perro, gato, mono.
- Series de Netflix, a menos que se extraigan métricas de visualización.

Path recomendado para encarar el TP

1. Hacé un script que consulte una API y lo cargue a una variable. (diccionario, pandas df, etc).
2. Subir esto a un repositorio de Github.
3. Agregar al script otra función que cargue esos datos al DW.
4. a. Diseñar e implementar el modelado de las tablas resultantes.
   b. Deployar Airflow con Docker.
5. Codear un DAG que corra los pasos de tu pipeline.
6. Añadir tests unitarios.
7. Añadir Github Actions que corran esos tests.
8. Crear PR que simule un cambio y corra tests.
9. Pulir algún requisito mínimo que no haya quedado del todo bien.
10. Continuar mejorando el tp añadiendo cualquier requisito mejora.

Se recomienda utilizar el espacio de clases de consulta para despejar dudas puntuales que te permitan avanzar con el resto de los requerimientos

ITBA

¿Qué podés utilizar para realizar este TP?

Hay múltiples herramientas que podés utilizar para guiarte en la realización del TP.

- Lo más obvio, **Google**. Cualquier tipo de búsqueda o documentación que encuentres, te puede servir.
- Cualquier video de **Youtube** para reforzar conocimientos. Podemos darte recomendaciones dependiendo el tema que no entiendas del todo o te sientas trabado.
- Las **clases grabadas** en el Campus.
- El **equipo de profes**, tanto por mail como Slack.

Herramientas que se pueden utilizar, con **responsabilidad** y en tanto te **ayuden a aprender** (de nada sirve que algo o alguien escriba código por vos):

- **ChatGPT**, Gemini o Claude, o cualquier otro modelo de IA. Recomendamos realizar prompts que nos permitan **aprender** y entender el código por nosotros sin entender qué se hace o **por qué.**

  Recomendamos fuertemente preguntarle al Chat que nos explique por qué un error ocurre, por qué una parte particular de nuestro código no funciona o qué recomienda hacer para hacer más eficiente nuestro código.

  No recomendamos pedirle a ChatGPT que nos escriba archivos completos desde 0.

- El Slack de forma comunitaria: si te sentís trabado con algo, no hay problema en preguntar en el Slack del curso. La colaboración y el trabajo de equipo es un **skill** que todo ingeniero necesita y lo hace crecer. Como con las IA, **NO se puede** preguntar cómo hacer las cosas desde 0, **SÍ se recomienda** preguntar por errores que uno ve en su entorno, o errores en código que no logras solucionar.

ITBA

¿Qué NO podés utilizar para realizar este TP?

El código de otra persona que esté realizando este TP o una consigna similar.

FAQ

¿Podré mostrarle este trabajo a un posible empleador?

Sí, la idea es que sea un punto más en tu CV. Normalmente, los empleadores nos solicitan nuestro Github para inspeccionar qué repositorios tenemos, y si tenemos proyectos personales. Recomendamos que **NO utilices palabras** clave como **ITBA, curso o TP**, principalmente para que **no sea fácilmente encontrable por otros próximos alumnos**, pero también para mostrarle al que quiera revisar nuestro repo que realizamos este trabajo por **cuenta propia** y no porque un curso te dijo que lo hagas.

Cosas que vimos que **hacen a la gente desaprobar**

1. **Procrastinar porque pensás que vas a sacar las cosas rápido.** Hay cosas que *no sabés que no sabés* hasta que llega el día que te sentás a hacer el TP y no te sale. **Date cuenta antes de que no era tan fácil y empezá a hacerlo apenas puedas, así podés estimar mejor cuánto te va a llevar.**
2. Confiar que los profes van a ignorar que **faltan 1 o 2 requisitos mínimos** en el TP y te van a aprobar igual. **NO, se llaman MÍNIMOS por algo. Te vamos a decir que re-entregues (a menos que ya no te quede tiempo).**
3. Quedarse 2 o 3 días trabados con algo en un requisito en particular. **Está bien que tenés semanas para hacerlo, pero si estás trabado, levantá la mano o anda a otro punto, sino perdiste.**
4. Hacer código horrible y no saber explicarlo, porque fue un copy paste con comentarios obvios salidos de ChatGPT.

Fechas y forma de entrega

Consigna dada: hoy 02/10/2025

Clases de consulta: 30/10/2025, 11/11/2025

Presentaciones de TPs terminados (o semi-terminados): 18/11/2025 (posible el 13/11 también).

Fecha de entrega sugerida: 18/11/2025 -> Si entregas antes de esta fecha, te asegurás al menos una instancia de feedback formal y el poder re-entregar antes de la fecha límite.

Fecha límite de entrega: 25/11/2025 23:59:59 (no hay más re-entregas).
Cargar link del repo a la actividad del Campus (y si pueden, mandenlo a los profes a Slack).

¿Cómo corregimos?

En la revisión de cualquier entrega, los requerimientos mínimos pueden tener 3 estados: nulo, insuficiente, aprobado.

Nulo: no está, no existe. Insuficiente: está pero le falta algo. Aprobado: req. está ok.

- Si un (1) o más requerimientos son **nulos** (inexistentes) -> TP ni revisado - sin feedback ❌.
- Si un (1) o más requerimientos son **insuficientes** -> TP para re-entrega ⚠️.
- Si todos los requerimientos mínimos están **aprobados** + **no llega a 5 puntos** de req.extra -> TP para re-entrega ⚠️.
- Si todos los requerimientos mínimos están **aprobados** + **5+ puntos** de reqs. extra -> TP aprobado ✅.

Referencias

Doc oficial de lanzar Airflow con Docker:
https://airflow.apache.org/docs/apache-airflow/2.11.0/howto/docker-compose/index.html

HAPPY CODING!