Airflows - Ejercicios

En esta práctica veremos cómo implementar workflows dinámicos en Airflow modificando ejercicios previos.

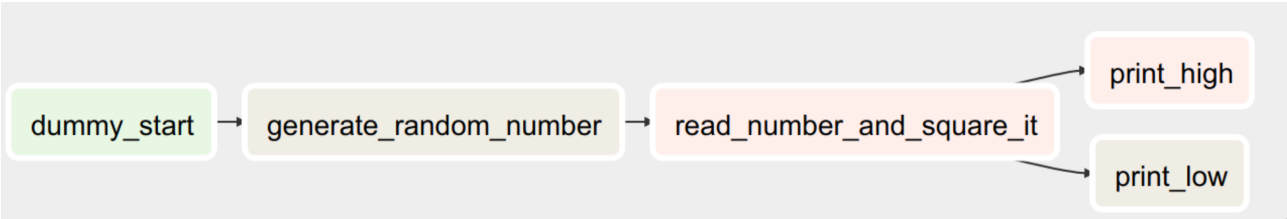
1. Ejercicio 1

Usando el archivo `airflow_tutorial.py`,

1. Cambiar el intervalo para que sea cada 30 minutos.
2. Use un sensor para agregar un delay de 5 minutos antes de comenzar.
3. Implementar el *template* para el `BashOperator`: imprima el `execution_date` en lugar de `'hello'` (Chequear el tutorial original y el ejemplo del DAG).
4. Implementar el *template* para el `PythonOperator` imprima el `execution_date` con una hora agregada en la función `print_world()` (Chequear la documentacion del `PythonOperator`).

(Extra) Ejercicio 2: Extendiendo el DAG de `Random_Number`

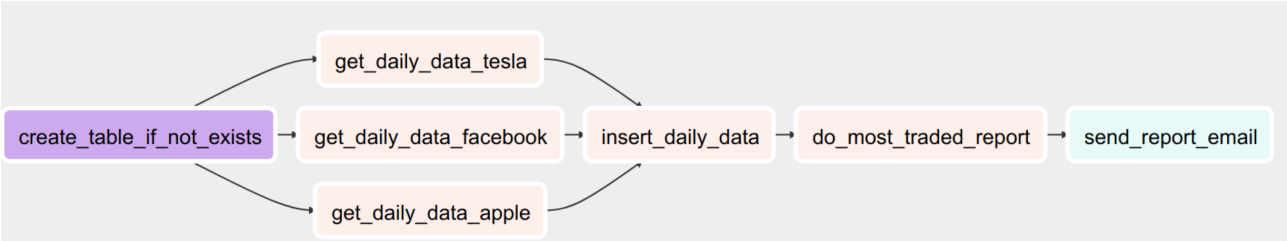
Usando el archivo `random_number_dag.py`, siga los siguientes pasos para implementar el flujo de trabajo que se muestra en la figura.



- Agregue mensajes de *logging* que indiquen a) qué archivo se va a leer, y b) qué número se leyó del archivo.
- El nombre del archivo está actualmente codificado como `random_number.txt`, cámbielo para mostrar la fecha de ejecución, es decir, los archivos ahora deberían llamarse así como `20181215.txt`. **Sugerencia:** debe capturar la fecha de ejecución agregando un ** argumento de contexto en el Python que llama.
- **Difícil:** Convierta el `PythonOperator` en un `BranchPythonOperator` que será seguido de otra tarea de `PythonOperator` (`BashOperator`) que imprime ALTA (BAJO) si el cuadrado resultante es mayor (menor) que 30.
- **Extra:** Agregue lógica para borrar los archivos temporales creados.

(Extra) Ejercicio 3: Extendiendo el DAG de Reporte de Mercados

Con el archivo `stock_dag.py`, realice los siguientes pasos para crear el *workflow* que se muestra en la figura.



- Modifique el *workflow* para obtener (e insertar) también datos de Tesla (tsla) y Facebook (fb) acciones.
  Nota: La forma fácil de hacer esto es agregar un bucle dentro del método que realiza la solicitud. El más difícil (visto en la figura) es tener una rama por empresa que se genera a partir de la tarea inicial que luego se reúne en la tarea de inserción.
- Tenga en cuenta que si borra una tarea de inserción y la vuelve a ejecutar, obtendrá un error (debido a la restricción única). Agregue un poco de lógica de prueba / captura para evitar este error en el insertar función de devolución de llamada.
  - Alternativamente, puede dejar que la tarea falle y agregar `trigger_rule = 'all_done'` a la tarea sucesiva para activarla independientemente del error de la tarea anterior.
- Agregar una nueva tarea que lea (es decir, consultas) todos los registros insertados en la ejecución actual y devuelve una cadena que indica cuál fue la empresa con el promedio de operaciones más alto por minuto en esa fecha.
- **Bonus:** Cree una tarea final que envíe un correo electrónico, utilizando EmailOperator, del informe / conocimiento obtenido en 3.
  Nota: Para este ejercicio, deberá modificar el entrada `smpt` en el `archivo airflow.cfg`