Authors: Brandon Jersaí Alfaro Checa, Eddie Conti

Assignment: Building an ETL Pipeline with Apache Airflow

Documentation ETL Pipeline

Functions

We first made a sketch on the structure of what our pipeline could look like, based on what was requested for this assignment: download the Online Retail dataset from UCI, preprocess (clean and transform) our data, and finally load it up into MongoDB. For that matter, we started defining regular Python functions for each potential step:

* download_dataset():

```
def download_dataset():
    url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Ret
ail.xlsx'
    output_path = os.path.join(data_dir, 'online_retail.xlsx')

    # Ensure the directory exists, if the directory does not exist, it will
be created
    os.makedirs(data_dir, exist_ok=True)

    # Download the dataset
    response = requests.get(url)

    # Check if download was successful
    if response.status_code == 200:
        with open(output_path, 'wb') as f:
            f.write(response.content)
        print(f"Dataset downloaded and saved to {output_path}")
    else:
        print(f"Failed to download dataset. Status code:
{response.status_code}")
```

Figure 1. Download Dataset Function

This first function downloads our desired dataset through a GET request, and saves it as an .xlsx file into the data_dir directory, which we defined to be created on the following path: data_dir = '/opt/airflow/dags/data'.

* clean_dataset():

```
def clean_dataset():
    input_path = os.path.join(data_dir, 'online_retail.xlsx')
    output_path = os.path.join(data_dir, 'cleaned_online_retail.csv')

    # Read xlsx file
    df = pd.read_excel(input_path)
    # Create a mapping dictionary to fill in missing Description values
    stockcode_description_map = (pd.read_excel(input_path)
                                 .groupby('StockCode')['Description']
                                 .agg(lambda x: x.mode().iloc[0] if not
x.empty and not x.mode().empty else None)
                                 .to_dict())

df['Description'].fillna(df['StockCode'].map(stockcode_description_map))

      # Convert data types
      df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
      df['CustomerID'] = df['CustomerID'].astype(str)

      # Remove duplicates
      df = df.drop_duplicates().reset_index(drop = True)
      # Save clean dataset
      df.to_csv(output_path, index=False)
      print("Data Cleaning completed successfully.")
```

Figure 2. Clean dataset function

This second function reads our downloaded dataset from the previous task, groups the data by 'StockCode', which identifies each unique product inside the dataset, and we compute which 'Description' (Product Name) is the most concurrent one for each 'StockCode'. The reasoning behind this calculation relies on filling empty records of 'Description' based on what seems to be the correct Description for each 'StockCode'. Note that there are inconsistencies on the data, and not every 'StockCode' has a unique product name in the 'Description' column (some StockCodes have additional notes on the product in 'Description' instead of the actual product name).

Afterwards we convert the 'InvoiceDate' to a datetime variable, and the 'CustomerID' column to a string variable, as we will not be performing numerical calculations with these IDs.

Finally, we drop duplicate records within our dataset and save this "cleansed" dataframe as a .csv file into our defined output_path.

* data_transformation():

```
def data_transformation():
    input_path = os.path.join(data_dir, 'cleaned_online_retail.csv')
    output_path = os.path.join(data_dir, 'transformed_online_retail.csv')

    # Read cleaned csv
    df = pd.read_csv(input_path)
    # Calculate Total Price
    df['TotalPrice'] = df['Quantity'] * df['UnitPrice']
    # Save transformed csv
    df.to_csv(output_path, index = False)
    print("Total Price column added successfully.")
```

Figure 3. Data transformation function

In this third function, we firstly read our cleansed dataset, and compute a new column, ‘TotalPrice’, which is quantified as the multiplication of the ‘Quantity’ of products in that order times the ‘UnitPrice’ of that specific product. Then we just saved this transformed dataframe into a .csv file.

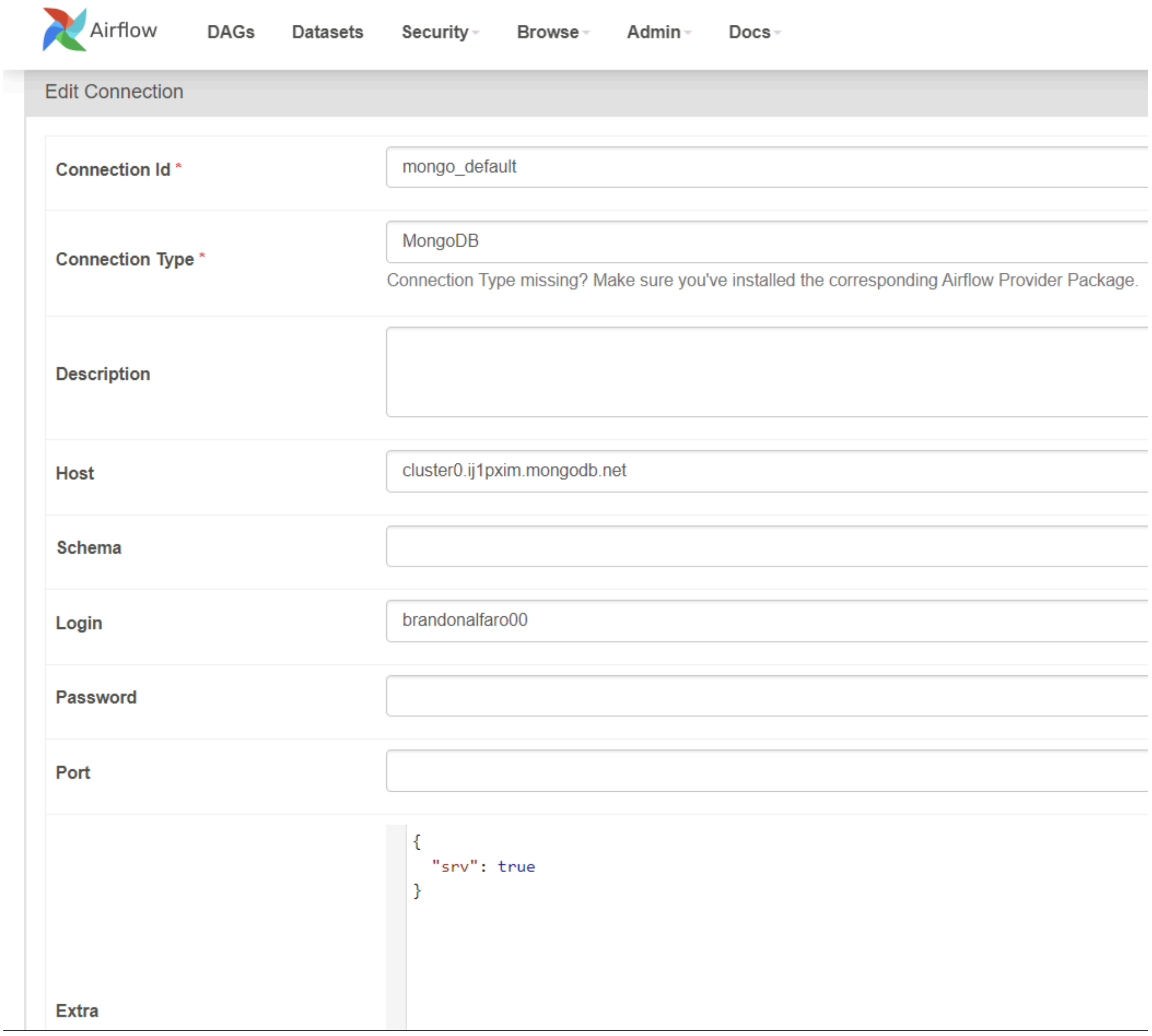
* load_to_mongodb():

```
def load_to_mongodb():
    # Establish connection using the MongoHook
    hook = MongoHook(mongo_conn_id='mongo_default')
    client = hook.get_conn()
    db = client.Online_Retail
    collection = db.Retail_Transactions
    print(f"Connected to MongoDB - {client.server_info()}")

    # Path to transformed dataset
    path = os.path.join(data_dir, 'transformed_online_retail.csv')
    df = pd.read_csv(path)

    # Create a unique index on the specified fields
    collection.create_index([
    ('InvoiceNo'),
    ('StockCode'),
    ('Description'),
    ('Quantity'),
    ('InvoiceDate'),
    ('UnitPrice'),
    ('CustomerID'),
    ('Country'),
    ('TotalPrice'),
    ], unique=True)

    # Split the DataFrame into chunks and insert each chunk separately (only
non-duplicate data)
    chunk_size = 10000
    total_documents_inserted = 0
    total_duplicates = 0
    for start in range(0, len(df), chunk_size):
        chunk = df.iloc[start : (start + chunk_size)]
        try:
            #ordered=False ensures that the insertion continues even if some
documents cause errors
            result = collection.insert_many(chunk.to_dict(orient='records'),
ordered=False)
            total_documents_inserted += len(result.inserted_ids)
            print(f"New documents inserted from chunk starting at row
{start}: {len(result.inserted_ids)}")
        except pymongo.errors.BulkWriteError as e:
            # Count duplicate documents in chunk
            duplicates_in_chunk = len(e.details['writeErrors'])
            total_duplicates += duplicates_in_chunk
            print(f"Duplicated documents in chunk starting at row {start}:
{duplicates_in_chunk}")
```

```
    print(f"Data insertion completed, new documents inserted 
{total_documents_inserted}, duplicated documents {total_duplicates}")
    return total_documents_inserted
```

Figure 4. Load to MongoDB function

In this last function, we are ready to load our cleaned and transformed dataset into MongoDB, in this case using MongoHook, which wraps the PyMongo Python Driver. In order for this to work, we first needed to create a Connection inside Airflow's UI (Admin -> Connections). Following the tutorial *Using MongoDB with Apache Airflow*, we first installed the MongoDB Airflow provider: apache-airflow-providers-mongo. Then, we created the following connection (Password is censored for safety measures):



Connection Id *	mongo_default
Connection Type *	MongoDB Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Host	cluster0.ij1pxim.mongodb.net
Schema	
Login	brandonalfaro00
Password	
Port	
Extra	{ "srv": true }

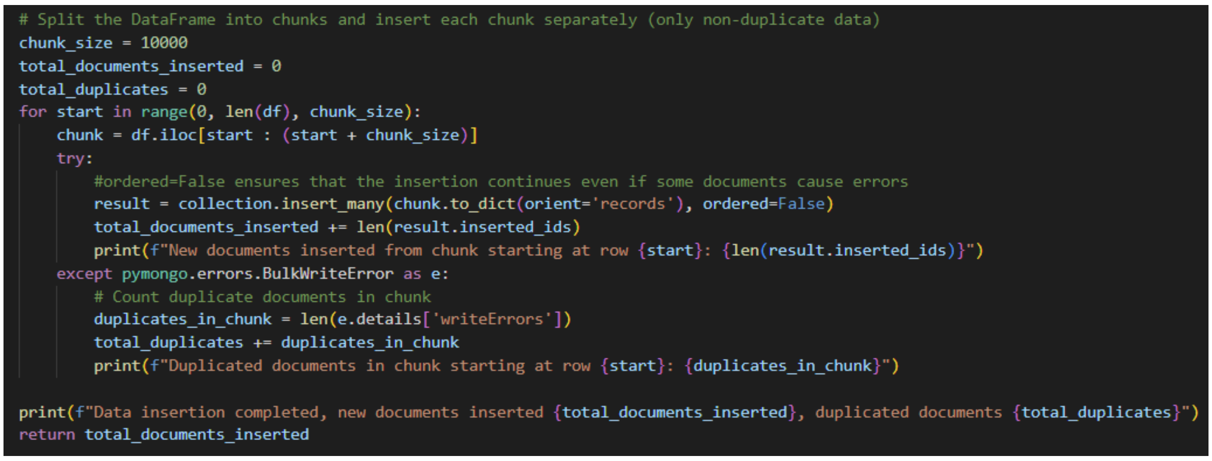
Figure 5. MongoDB Connection

In this connection, Host refers to our MongoDB Atlas hostname; Login and Password refers to our database username and password, respectively; and Extra {"srv": true}, which eliminates the requirement for every client to pass in a complete set of state information for the cluster.

Following up with our function, we firstly establish the connection to MongoDB using MongoHook and the ConnectionID, and create our database and collection, where we'll store our dataset. After that, we defined an unique index for each combination of values, considering all of our columns. This would prevent adding duplicate documents into our database.

Now, since our dataset is too large, we would need to insert documents by chunks instead of the whole set of documents at once, since the maximum BSON document size is 16 mb (we tried using collection.insert_many for all of the documents and got this error).

By chunks of 10,000 documents, all documents were finally added into the database. Note that we included a try/except block so all new documents could be added into db, but duplicated documents. Within each chunk, the number of new documents added to the database will be printed (len(result.inserted_ids), as well as the number of documents that were already loaded into it (duplicates_in_chunk):

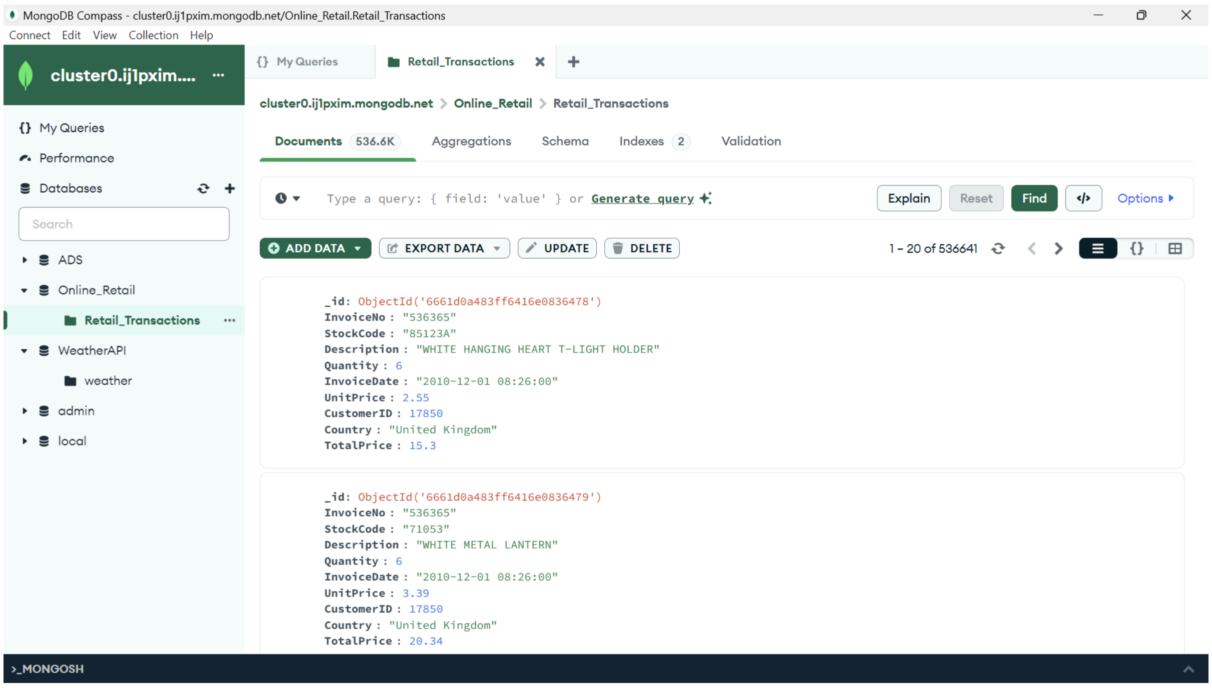


```python
# Split the DataFrame into chunks and insert each chunk separately (only non-duplicate data)
chunk_size = 10000
total_documents_inserted = 0
total_duplicates = 0
for start in range(0, len(df), chunk_size):
    chunk = df.iloc[start : (start + chunk_size)]
    try:
        #ordered=False ensures that the insertion continues even if some documents cause errors
        result = collection.insert_many(chunk.to_dict(orient='records'), ordered=False)
        total_documents_inserted += len(result.inserted_ids)
        print(f"New documents inserted from chunk starting at row {start}: {len(result.inserted_ids)}")
    except pymongo.errors.BulkWriteError as e:
        # Count duplicate documents in chunk
        duplicates_in_chunk = len(e.details['writeErrors'])
        total_duplicates += duplicates_in_chunk
        print(f"Duplicated documents in chunk starting at row {start}: {duplicates_in_chunk}")

print(f"Data insertion completed, new documents inserted {total_documents_inserted}, duplicated documents {total_duplicates}")
return total_documents_inserted
```

Figure 7. Load to MongoDB function - dealing with duplicate docs

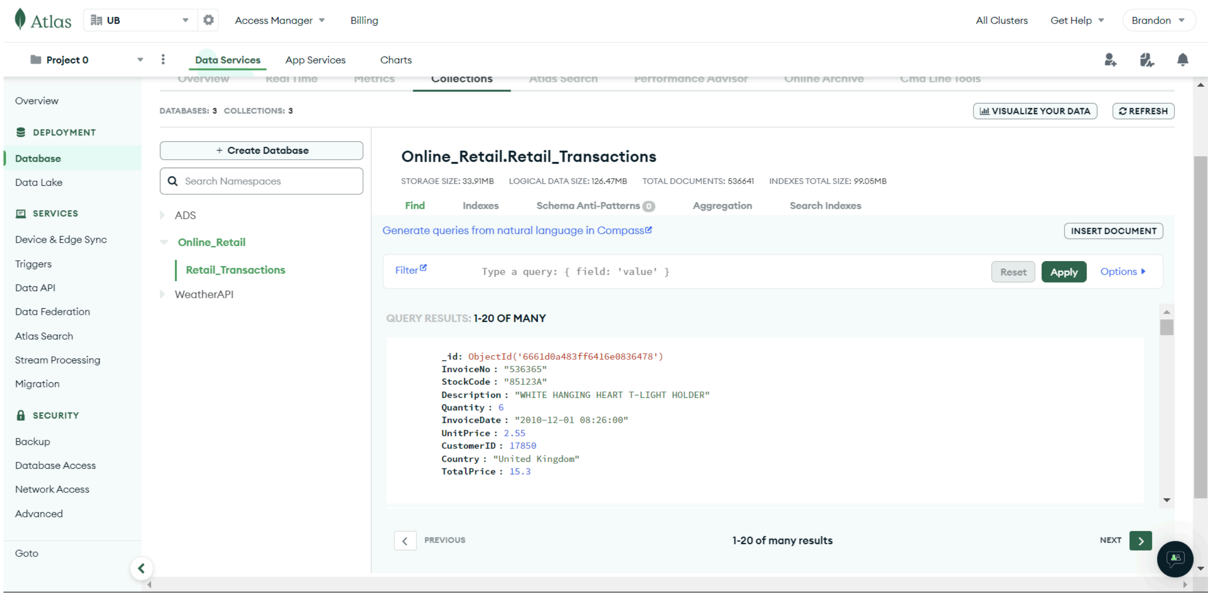
Skipping documents that are already loaded in the database, instead of adding duplicate documents is very helpful in day-to-day real world scenarios.
This would happen if, say, the Online Retail dataset was continuously updated with new transactions, obviously we would not like past transactions to be recorded into the db since we already have them loaded.

After uploading all 536,641 documents, this is how our NoSQL database looks like:



```
_id: ObjectId('6661d0a483ff6416e0836478')
InvoiceNo : "536365"
StockCode : "85123A"
Description : "WHITE HANGING HEART T-LIGHT HOLDER"
Quantity : 6
InvoiceDate : "2010-12-01 08:26:00"
UnitPrice : 2.55
CustomerID : 17850
Country : "United Kingdom"
TotalPrice : 15.3

_id: ObjectId('6661d0a483ff6416e0836479')
InvoiceNo : "536365"
StockCode : "71053"
Description : "WHITE METAL LANTERN"
Quantity : 6
InvoiceDate : "2010-12-01 08:26:00"
UnitPrice : 3.39
CustomerID : 17850
Country : "United Kingdom"
TotalPrice : 20.34
```

Figure 8. NoSQL database with loaded documents - MongoDB Compass View



```
_id: ObjectId('6661d0a483ff6416e0836478')
InvoiceNo : "536365"
StockCode : "85123A"
Description : "WHITE HANGING HEART T-LIGHT HOLDER"
Quantity : 6
InvoiceDate : "2010-12-01 08:26:00"
UnitPrice : 2.55
CustomerID : 17850
Country : "United Kingdom"
TotalPrice : 15.3
```

Figure 9. NoSQL database with loaded documents - MongoDB Atlas View

Inside the RetailTransactions.py file you can also find a version of load_to_mongodb() using only pymongo, with the traditional way of connecting from a Python script to MongoDB.

Tasks

After defining these functions, we now can delve into the creation of tasks and their dependencies inside the DAG.

```
# ======================== TASKS =========================
# Task 1: Download data
download_task = PythonOperator(
    task_id='download_dataset',
    python_callable=download_dataset,
    dag=dag
)

# Task 2: Clean data
clean_task = PythonOperator(
    task_id='clean_dataset',
    python_callable=clean_dataset,
    dag=dag
)

# Task 3: Transform data
transformation_task = PythonOperator(
    task_id='data_transformation',
    python_callable=data_transformation,
    dag=dag
)

# Task 4: Load data into MongoDB
load_to_mongodb_task = PythonOperator(
    task_id='load_to_mongodb',
    python_callable=load_to_mongodb,
    dag=dag
)
```

Figure 10. DAG Tasks

As you can see, this links each one of our functions to a task. Then, dependencies or the order on how tasks will be run can be seen here:

```
download_task >> clean_task >> transformation_task >> load_to_mongodb_task
```

Figure 11. DAG Tasks Dependencies

Finally, we would just need to explain what the creation of the DAG looks like. We named it as online_retail_etl, specifying that this DAG should be run @daily, which makes sure the DAG runs once a day at midnight:

```
# ========================== DAG ==========================
dag = DAG('online_retail_etl',
    default_args=default_args,
    description='ETL pipeline for Online Retail dataset',
    schedule_interval='@daily',  # Run the task every midnight
    catchup=False,  # Do not catch up on missing DAG runs
)
```

Figure 12. DAG

With all these, we successfully completed the required ETL process. Nonetheless, we decided to add an extra bonus to this process: send an email notification on success:

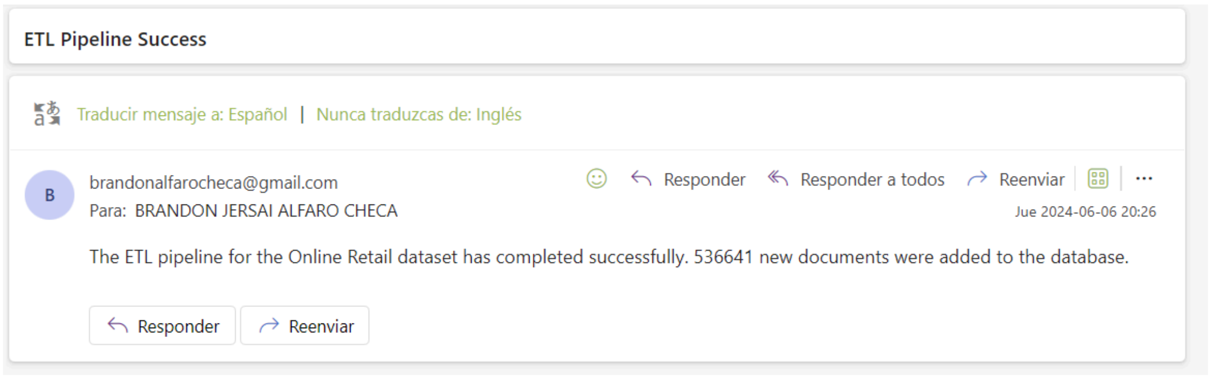
```
# Task 5: Email notification on success
success_email_task = EmailOperator(
    task_id='send_email_on_success',
    to='bralfarc7@alumnes.ub.edu',
    subject='ETL Pipeline Success',
    html_content=
    '''The ETL pipeline for the Online Retail dataset has completed
successfully.
    {{ task_instance.xcom_pull(task_ids="load_to_mongodb") }} new documents
were added to the database.''',
    dag=dag
)
```

Figure 13. Task: Email notification on success

```
download_task >> clean_task >> transformation_task >> load_to_mongodb_task >>
success_email_task
```

Figure 14. Task dependencies with email notification on success

If all of the previous tasks were successful, an email will be sent to bralfacr7@alumnes.ub.edu with the subject "ETL Pipeline Success" and the message "The ETL pipeline for the Online Retail dataset has completed successfully", also adding the total number of new documents that were loaded into our database (this was possible due to Xcom, a mechanism that lets tasks communicate with each other, this let us pass info from 'load_to_mongodb' task to 'success_email_task) :



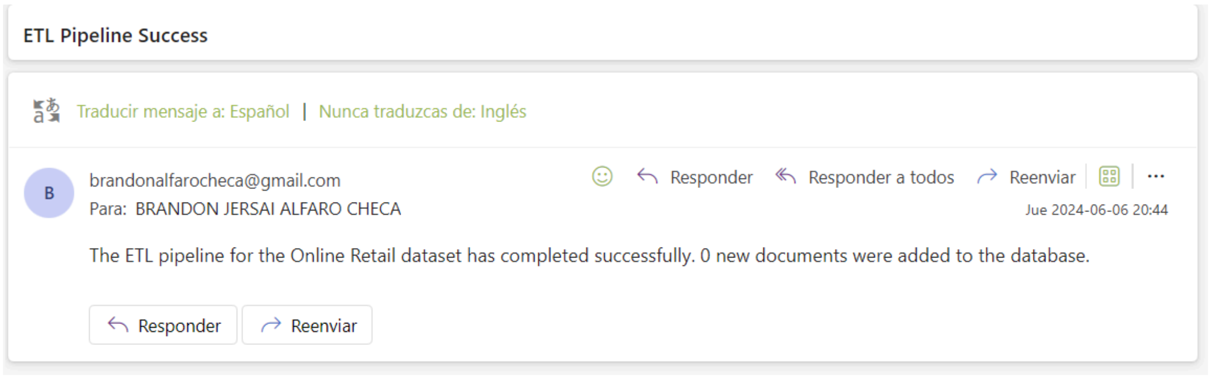
ETL Pipeline Success

Traducir mensaje a: Español | Nunca traduzcas de: Inglés

brandonalfarocheca@gmail.com
Para: BRANDON JERSAI ALFARO CHECA
Jue 2024-06-06 20:26

The ETL pipeline for the Online Retail dataset has completed successfully. 536641 new documents were added to the database.

Figure 15. Email notification on success



ETL Pipeline Success

Traducir mensaje a: Español | Nunca traduzcas de: Inglés

brandonalfarocheca@gmail.com
Para: BRANDON JERSAI ALFARO CHECA
Jue 2024-06-06 20:44

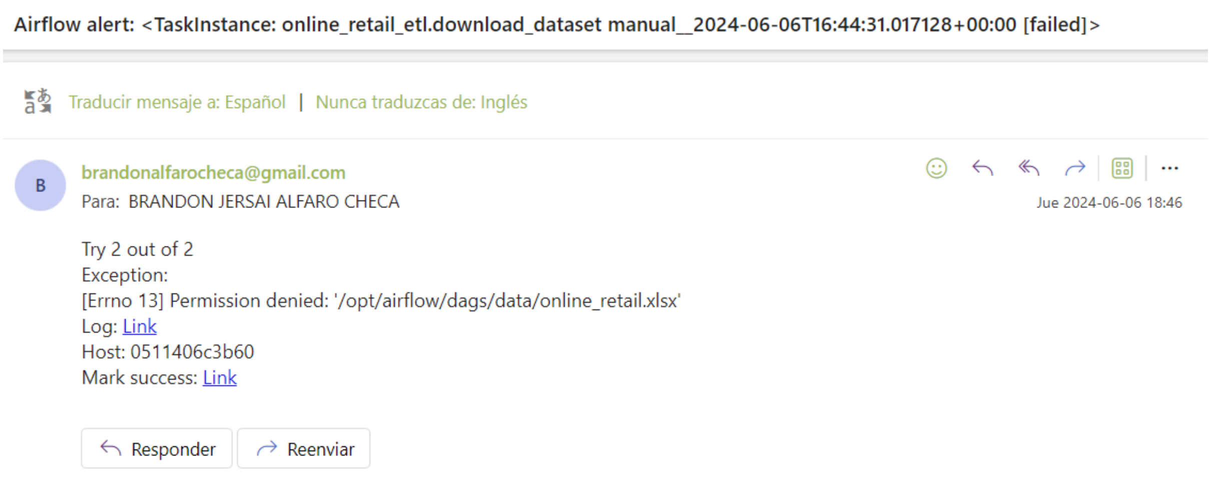
The ETL pipeline for the Online Retail dataset has completed successfully. 0 new documents were added to the database.

Figure 16. Email notification on success (no new docs added)

Finally let us take a look into the default_args that were passed on into the creation of our DAG. We set up just one retry per failed task, with a minute between retries.
Also note that email notifications will be also sent to that same email address if any of the tasks fail (email_on_failure = True).

```
default_args = {
    'owner': 'Brandon and Eddie',
    'depends_on_past': False,  #  Tasks will run regardless of the status of the same task in the previous DAG run
    'start_date': datetime(2024, 6, 1),
    'email': 'bralfarc7@alumnes.ub.edu',
    'email_on_failure': True, # Send email on failure
    'email_on_retry': False, # Do not send email on retry
    'retries': 1, # Number of retries
    'retry_delay': timedelta(minutes=1) # Time between retries
}
```

Figure 17. Default Arguments for DAG initialization


Airflow alert: <TaskInstance: online_retail_etl.download_dataset manual__2024-06-06T16:44:31.017128+00:00 [failed]>

brandonalfarocheca@gmail.com
Para: BRANDON JERSAI ALFARO CHECA

```
Try 2 out of 2
Exception:
[Errno 13] Permission denied: '/opt/airflow/dags/data/online_retail.xlsx'
Log: Link
Host: 0511406c3b60
Mark success: Link
```

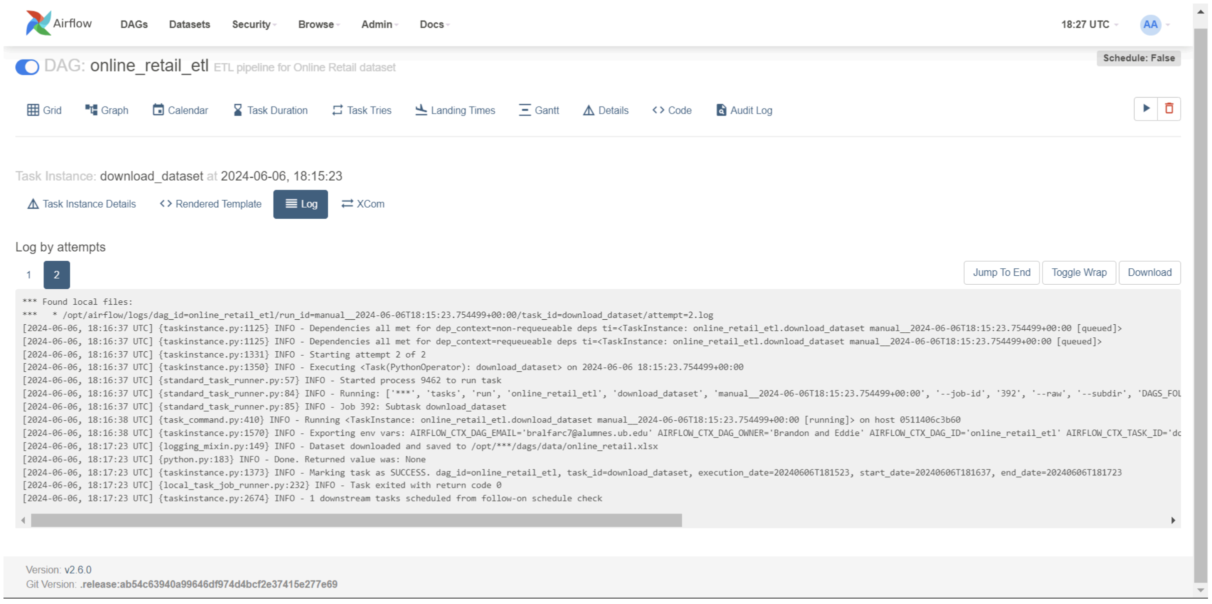
Figure 18. Task failure alert via email notification

(In this case I had the .xlsx opened on Excel, which prevented any modifications and hence the task download_dataset failed.)

Authors: Brandon Jersaí Alfaro Checa, Eddie Conti

Logs Execution

* Task 1: download_dataset



DAG: online_retail_etl ETL pipeline for Online Retail dataset

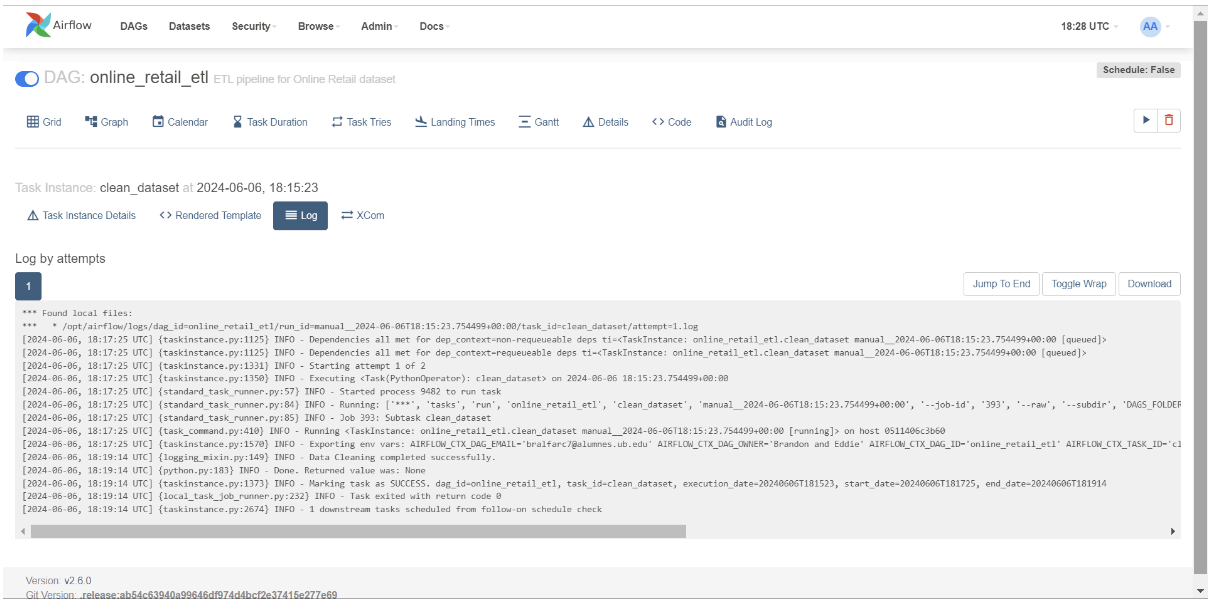
Task Instance: download_dataset at 2024-06-06, 18:15:23

Log by attempts

```
*** Found local files:
***   * /opt/airflow/logs/dag_id=online_retail_etl/run_id=manual__2024-06-06T18:15:23.754499+00:00/task_id=download_dataset/attempt=2.log
[2024-06-06, 18:16:37 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: online_retail_etl.download_dataset manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:16:37 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: online_retail_etl.download_dataset manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:16:37 UTC] {taskinstance.py:1331} INFO - Starting attempt 2 of 2
[2024-06-06, 18:16:37 UTC] {taskinstance.py:1350} INFO - Executing <Task(PythonOperator): download_dataset> on 2024-06-06 18:15:23.754499+00:00
[2024-06-06, 18:16:37 UTC] {standard_task_runner.py:57} INFO - Started process 9462 to run task
[2024-06-06, 18:16:37 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'online_retail_etl', 'download_dataset', 'manual__2024-06-06T18:15:23.754499+00:00', '--job-id', '392', '--raw', '--subdir', 'DAGS_FOL
[2024-06-06, 18:16:37 UTC] {standard_task_runner.py:85} INFO - Job 392: Subtask download_dataset
[2024-06-06, 18:16:38 UTC] {task_command.py:410} INFO - Running <TaskInstance: online_retail_etl.download_dataset manual__2024-06-06T18:15:23.754499+00:00 [running]> on host 0511406c3b60
[2024-06-06, 18:16:38 UTC] {taskinstance.py:1570} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='bralfarc7@alumnes.ub.edu' AIRFLOW_CTX_DAG_OWNER='Brandon and Eddie' AIRFLOW_CTX_DAG_ID='online_retail_etl' AIRFLOW_CTX_TASK_ID='dc
[2024-06-06, 18:17:23 UTC] {logging_mixin.py:149} INFO - Dataset downloaded and saved to /opt/***/dags/data/online_retail.xlsx
[2024-06-06, 18:17:23 UTC] {python.py:183} INFO - Done. Returned value was: None
[2024-06-06, 18:17:23 UTC] {taskinstance.py:1373} INFO - Marking task as SUCCESS. dag_id=online_retail_etl, task_id=download_dataset, execution_date=20240606T181523, start_date=20240606T181637, end_date=20240606T181723
[2024-06-06, 18:17:23 UTC] {local_task_job_runner.py:232} INFO - Task exited with return code 0
[2024-06-06, 18:17:23 UTC] {taskinstance.py:2674} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

Version: v2.6.0
Git Version: .release:ab54c63940a99646df974d4bcf2e37415e277e69

* Task 2: clean_dataset



DAG: online_retail_etl ETL pipeline for Online Retail dataset

Task Instance: clean_dataset at 2024-06-06, 18:15:23

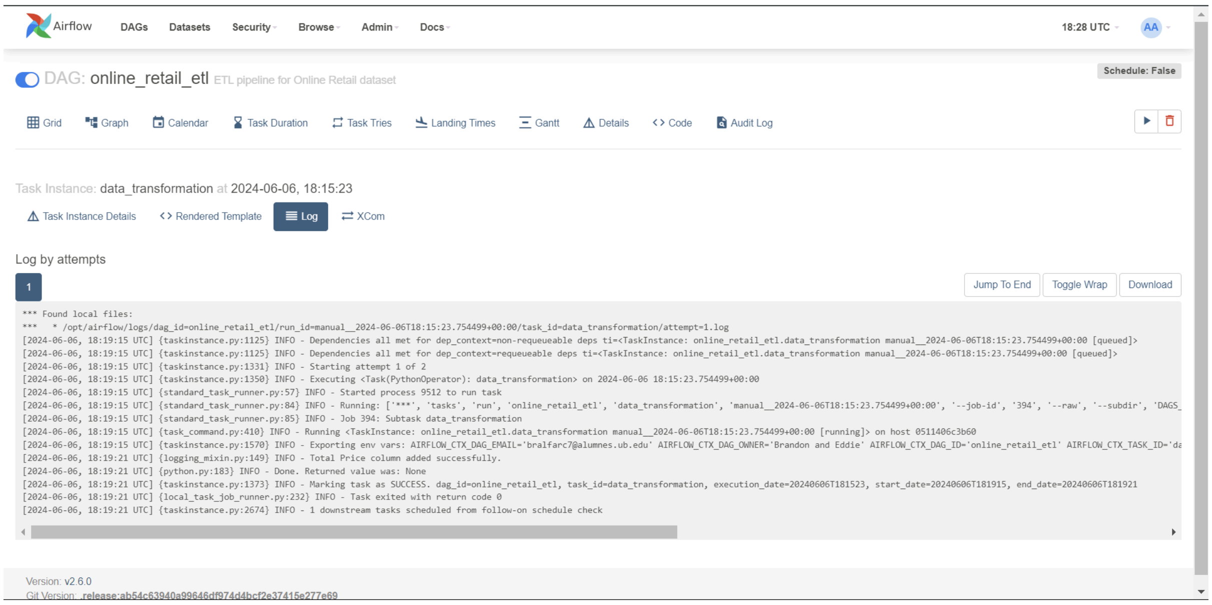
Log by attempts

```
*** Found local files:
***   * /opt/airflow/logs/dag_id=online_retail_etl/run_id=manual__2024-06-06T18:15:23.754499+00:00/task_id=clean_dataset/attempt=1.log
[2024-06-06, 18:17:25 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: online_retail_etl.clean_dataset manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:17:25 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: online_retail_etl.clean_dataset manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:17:25 UTC] {taskinstance.py:1331} INFO - Starting attempt 1 of 2
[2024-06-06, 18:17:25 UTC] {taskinstance.py:1350} INFO - Executing <Task(PythonOperator): clean_dataset> on 2024-06-06 18:15:23.754499+00:00
[2024-06-06, 18:17:25 UTC] {standard_task_runner.py:57} INFO - Started process 9482 to run task
[2024-06-06, 18:17:25 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'online_retail_etl', 'clean_dataset', 'manual__2024-06-06T18:15:23.754499+00:00', '--job-id', '393', '--raw', '--subdir', 'DAGS_FOLDER
[2024-06-06, 18:17:25 UTC] {standard_task_runner.py:85} INFO - Job 393: Subtask clean_dataset
[2024-06-06, 18:17:25 UTC] {task_command.py:410} INFO - Running <TaskInstance: online_retail_etl.clean_dataset manual__2024-06-06T18:15:23.754499+00:00 [running]> on host 0511406c3b60
[2024-06-06, 18:17:25 UTC] {taskinstance.py:1570} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='bralfarc7@alumnes.ub.edu' AIRFLOW_CTX_DAG_OWNER='Brandon and Eddie' AIRFLOW_CTX_DAG_ID='online_retail_etl' AIRFLOW_CTX_TASK_ID='cl
[2024-06-06, 18:19:14 UTC] {logging_mixin.py:149} INFO - Data Cleaning completed successfully.
[2024-06-06, 18:19:14 UTC] {python.py:183} INFO - Done. Returned value was: None
[2024-06-06, 18:19:14 UTC] {taskinstance.py:1373} INFO - Marking task as SUCCESS. dag_id=online_retail_etl, task_id=clean_dataset, execution_date=20240606T181523, start_date=20240606T181725, end_date=20240606T181914
[2024-06-06, 18:19:14 UTC] {local_task_job_runner.py:232} INFO - Task exited with return code 0
[2024-06-06, 18:19:14 UTC] {taskinstance.py:2674} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

Version: v2.6.0
Git Version: .release:ab54c63940a99646df974d4bcf2e37415e277e69

Authors: Brandon Jersaí Alfaro Checa, Eddie Conti

* Task 3: data_transformation



DAG: online_retail_etl ETL pipeline for Online Retail dataset

Task Instance: data_transformation at 2024-06-06, 18:15:23

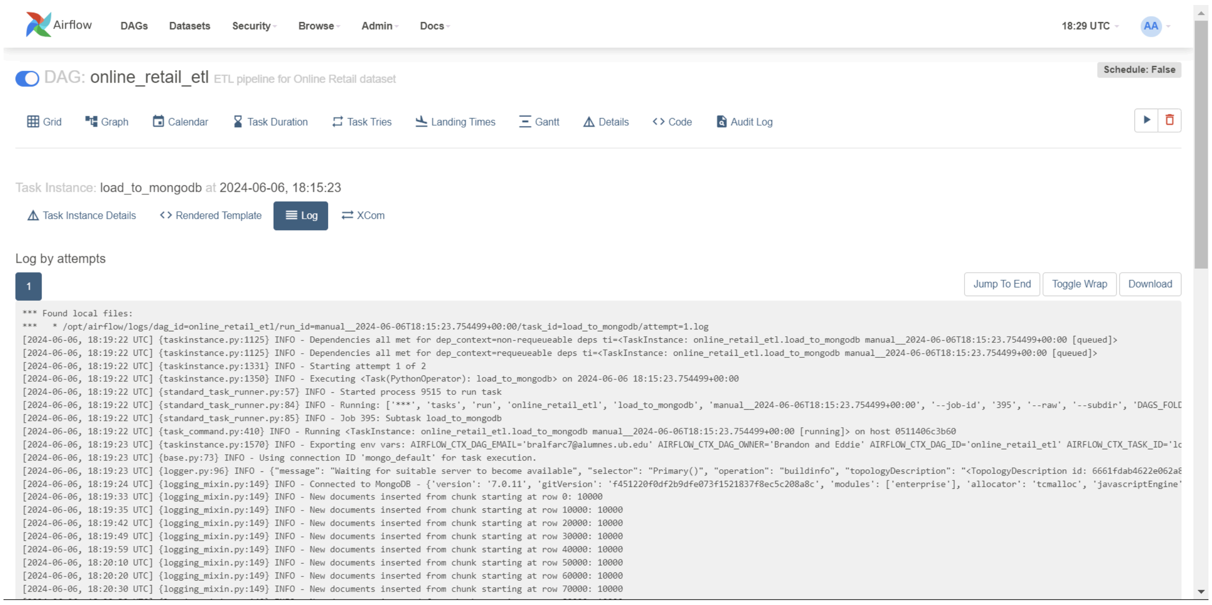
Log by attempts

1

```
*** Found local files:
***   * /opt/airflow/logs/dag_id=online_retail_etl/run_id=manual__2024-06-06T18:15:23.754499+00:00/task_id=data_transformation/attempt=1.log
[2024-06-06, 18:19:15 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: online_retail_etl.data_transformation manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:19:15 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: online_retail_etl.data_transformation manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:19:15 UTC] {taskinstance.py:1331} INFO - Starting attempt 1 of 2
[2024-06-06, 18:19:15 UTC] {taskinstance.py:1350} INFO - Executing <Task(PythonOperator): data_transformation> on 2024-06-06 18:15:23.754499+00:00
[2024-06-06, 18:19:15 UTC] {standard_task_runner.py:57} INFO - Started process 9512 to run task
[2024-06-06, 18:19:15 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'online_retail_etl', 'data_transformation', 'manual__2024-06-06T18:15:23.754499+00:00', '--job-id', '394', '--raw', '--subdir', 'DAGS_
[2024-06-06, 18:19:15 UTC] {standard_task_runner.py:85} INFO - Job 394: Subtask data_transformation
[2024-06-06, 18:19:15 UTC] {task_command.py:410} INFO - Running <TaskInstance: online_retail_etl.data_transformation manual__2024-06-06T18:15:23.754499+00:00 [running]> on host 0511406c3b60
[2024-06-06, 18:19:15 UTC] {taskinstance.py:1570} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='bralfarc7@alumnes.ub.edu' AIRFLOW_CTX_DAG_OWNER='Brandon and Eddie' AIRFLOW_CTX_DAG_ID='online_retail_etl' AIRFLOW_CTX_TASK_ID='da
[2024-06-06, 18:19:21 UTC] {logging_mixin.py:149} INFO - Total Price column added successfully.
[2024-06-06, 18:19:21 UTC] {python.py:183} INFO - Done. Returned value was: None
[2024-06-06, 18:19:21 UTC] {taskinstance.py:1373} INFO - Marking task as SUCCESS. dag_id=online_retail_etl, task_id=data_transformation, execution_date=20240606T181523, start_date=20240606T181915, end_date=20240606T181921
[2024-06-06, 18:19:21 UTC] {local_task_job_runner.py:232} INFO - Task exited with return code 0
[2024-06-06, 18:19:21 UTC] {taskinstance.py:2674} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

Version: v2.6.0
Git Version: .release:ab54c63940a99646df974d4bcf2e37415e277e69

* Task 4: load_to_mongodb



DAG: online_retail_etl ETL pipeline for Online Retail dataset

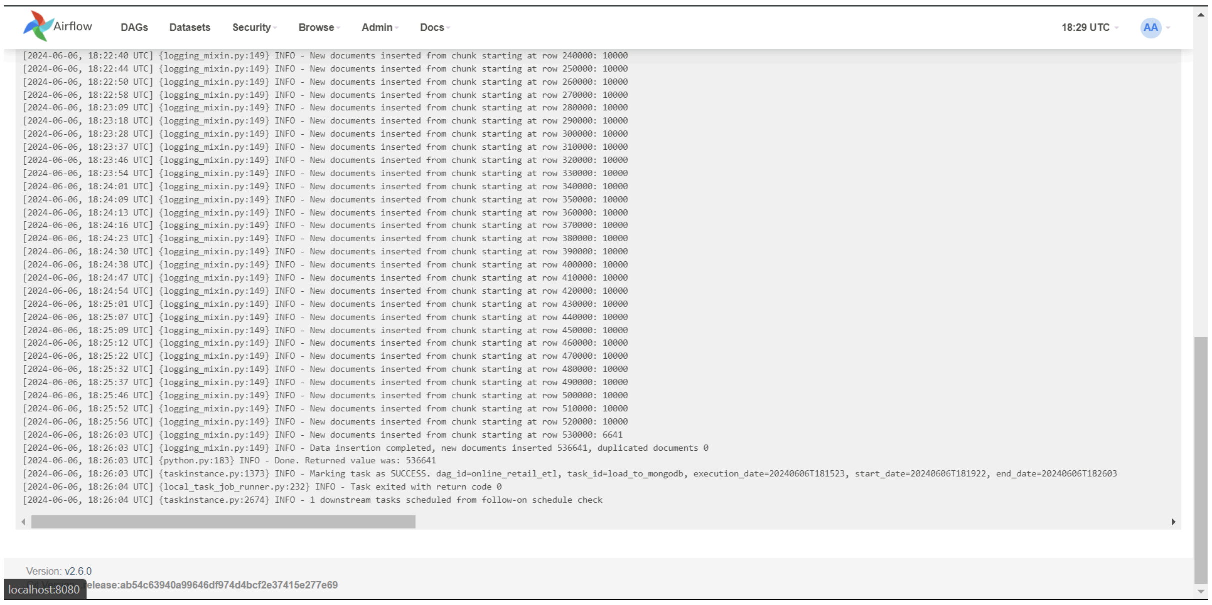
Task Instance: load_to_mongodb at 2024-06-06, 18:15:23

Log by attempts

1

```
*** Found local files:
***   * /opt/airflow/logs/dag_id=online_retail_etl/run_id=manual__2024-06-06T18:15:23.754499+00:00/task_id=load_to_mongodb/attempt=1.log
[2024-06-06, 18:19:22 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: online_retail_etl.load_to_mongodb manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:19:22 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: online_retail_etl.load_to_mongodb manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:19:22 UTC] {taskinstance.py:1331} INFO - Starting attempt 1 of 2
[2024-06-06, 18:19:22 UTC] {taskinstance.py:1350} INFO - Executing <Task(PythonOperator): load_to_mongodb> on 2024-06-06 18:15:23.754499+00:00
[2024-06-06, 18:19:22 UTC] {standard_task_runner.py:57} INFO - Started process 9515 to run task
[2024-06-06, 18:19:22 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'online_retail_etl', 'load_to_mongodb', 'manual__2024-06-06T18:15:23.754499+00:00', '--job-id', '395', '--raw', '--subdir', 'DAGS_FOLD
[2024-06-06, 18:19:22 UTC] {standard_task_runner.py:85} INFO - Job 395: Subtask load_to_mongodb
[2024-06-06, 18:19:22 UTC] {task_command.py:410} INFO - Running <TaskInstance: online_retail_etl.load_to_mongodb manual__2024-06-06T18:15:23.754499+00:00 [running]> on host 0511406c3b60
[2024-06-06, 18:19:23 UTC] {taskinstance.py:1570} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='bralfarc7@alumnes.ub.edu' AIRFLOW_CTX_DAG_OWNER='Brandon and Eddie' AIRFLOW_CTX_DAG_ID='online_retail_etl' AIRFLOW_CTX_TASK_ID='lo
[2024-06-06, 18:19:23 UTC] {base.py:73} INFO - Using connection ID 'mongo_default' for task execution.
[2024-06-06, 18:19:23 UTC] {logger.py:96} INFO - {"message": "Waiting for suitable server to become available", "selector": "Primary()", "operation": "buildinfo", "topologyDescription": "<TopologyDescription id: 6661fdab4622e062a8
[2024-06-06, 18:19:24 UTC] {logging_mixin.py:149} INFO - Connected to MongoDB - {'version': '7.0.11', 'gitVersion': 'f451220f0df2b9dfe073f1521837f8ec5c208a8c', 'modules': ['enterprise'], 'allocator': 'tcmalloc', 'javascriptEngine'
[2024-06-06, 18:19:33 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 0: 10000
[2024-06-06, 18:19:35 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 10000: 10000
[2024-06-06, 18:19:42 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 20000: 10000
[2024-06-06, 18:19:49 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 30000: 10000
[2024-06-06, 18:19:59 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 40000: 10000
[2024-06-06, 18:20:10 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 50000: 10000
[2024-06-06, 18:20:20 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 60000: 10000
[2024-06-06, 18:20:30 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 70000: 10000
```

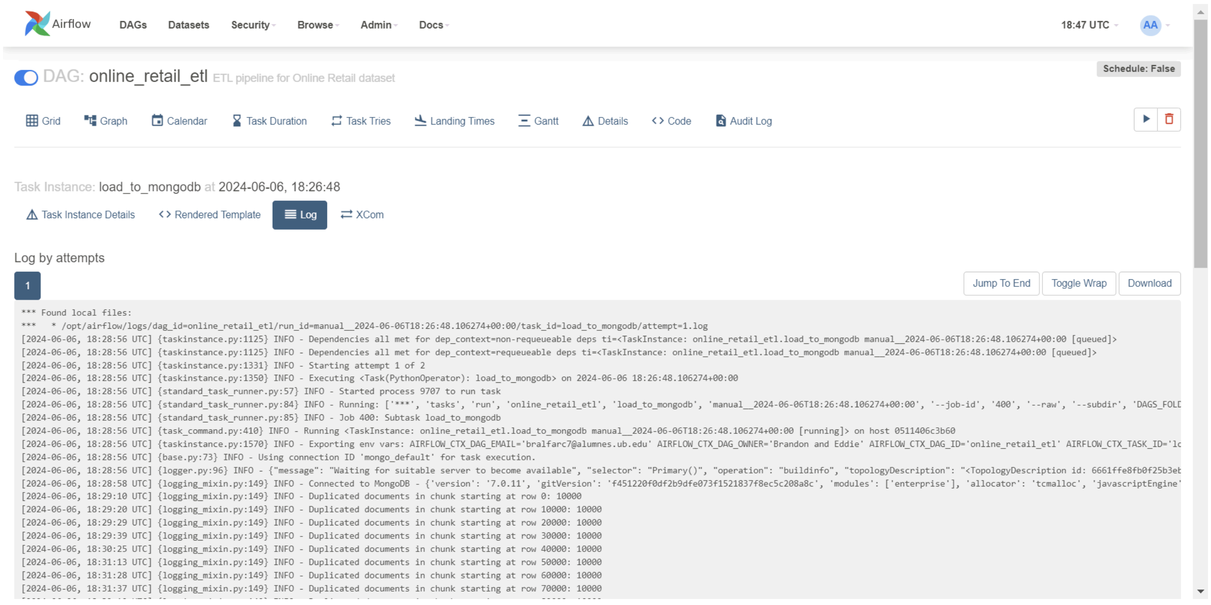
Authors: Brandon Jersaí Alfaro Checa, Eddie Conti



```
[2024-06-06, 18:22:40 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 240000: 10000
[2024-06-06, 18:22:44 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 250000: 10000
[2024-06-06, 18:22:50 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 260000: 10000
[2024-06-06, 18:22:58 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 270000: 10000
[2024-06-06, 18:23:09 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 280000: 10000
[2024-06-06, 18:23:18 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 290000: 10000
[2024-06-06, 18:23:28 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 300000: 10000
[2024-06-06, 18:23:37 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 310000: 10000
[2024-06-06, 18:23:46 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 320000: 10000
[2024-06-06, 18:23:54 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 330000: 10000
[2024-06-06, 18:24:01 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 340000: 10000
[2024-06-06, 18:24:09 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 350000: 10000
[2024-06-06, 18:24:13 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 360000: 10000
[2024-06-06, 18:24:16 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 370000: 10000
[2024-06-06, 18:24:23 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 380000: 10000
[2024-06-06, 18:24:30 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 390000: 10000
[2024-06-06, 18:24:38 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 400000: 10000
[2024-06-06, 18:24:47 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 410000: 10000
[2024-06-06, 18:24:54 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 420000: 10000
[2024-06-06, 18:25:01 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 430000: 10000
[2024-06-06, 18:25:07 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 440000: 10000
[2024-06-06, 18:25:09 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 450000: 10000
[2024-06-06, 18:25:12 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 460000: 10000
[2024-06-06, 18:25:22 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 470000: 10000
[2024-06-06, 18:25:32 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 480000: 10000
[2024-06-06, 18:25:37 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 490000: 10000
[2024-06-06, 18:25:46 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 500000: 10000
[2024-06-06, 18:25:52 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 510000: 10000
[2024-06-06, 18:25:56 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 520000: 10000
[2024-06-06, 18:26:03 UTC] {logging_mixin.py:149} INFO - New documents inserted from chunk starting at row 530000: 6641
[2024-06-06, 18:26:03 UTC] {logging_mixin.py:149} INFO - Data insertion completed, new documents inserted 536641, duplicated documents 0
[2024-06-06, 18:26:03 UTC] {python.py:183} INFO - Done. Returned value was: 536641
[2024-06-06, 18:26:03 UTC] {taskinstance.py:1373} INFO - Marking task as SUCCESS. dag_id=online_retail_etl, task_id=load_to_mongodb, execution_date=20240606T181523, start_date=20240606T181922, end_date=20240606T182603
[2024-06-06, 18:26:04 UTC] {local_task_job_runner.py:232} INFO - Task exited with return code 0
[2024-06-06, 18:26:04 UTC] {taskinstance.py:2674} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

Version: v2.6.0
localhost:8080 elease:ab54c63940a99646df974d4bcf2e37415e277e69

`* Task 4B: load_to_mongodb (case when we have past documents loaded already into database)`

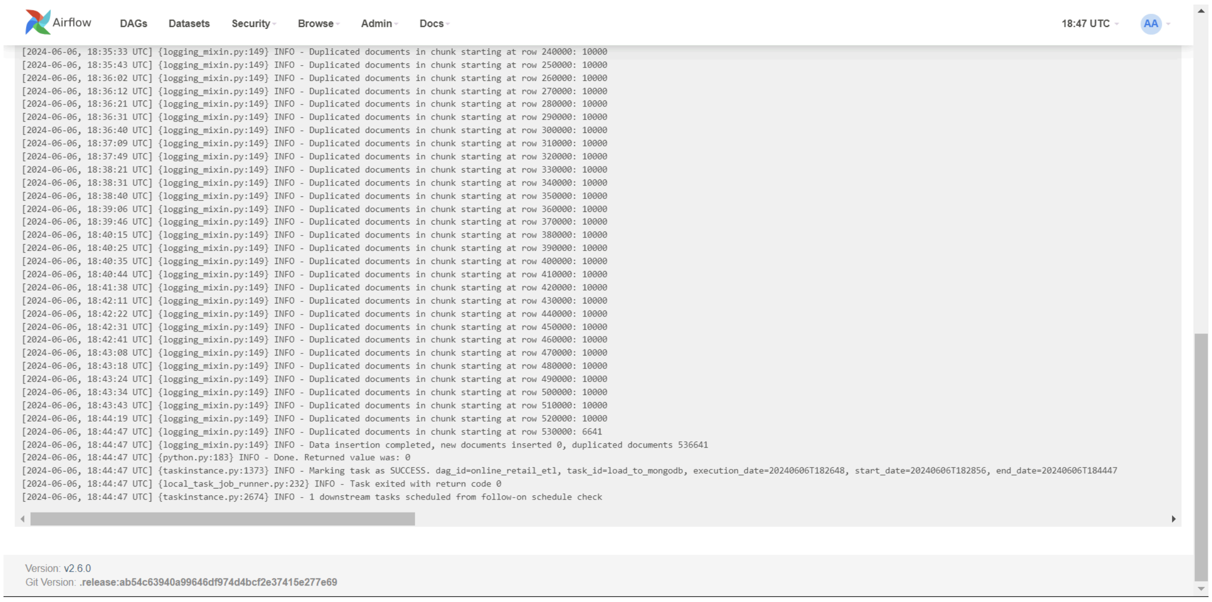


DAG: online_retail_etl ETL pipeline for Online Retail dataset

Task Instance: load_to_mongodb at 2024-06-06, 18:26:48

Log by attempts

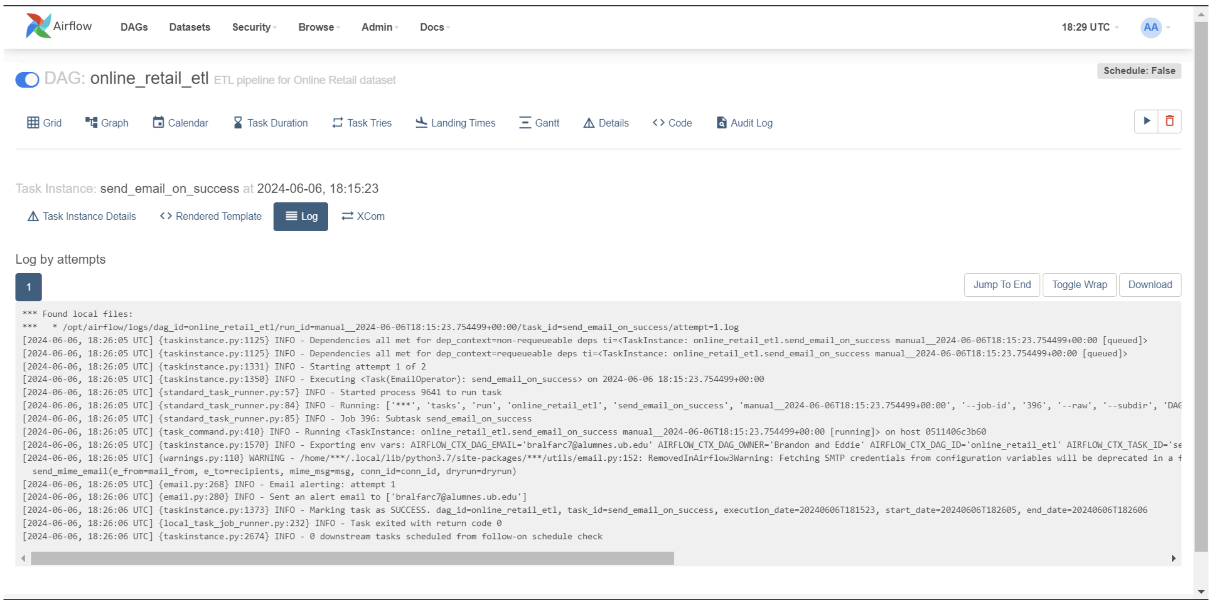
```
*** Found local files:
***   * /opt/airflow/logs/dag_id=online_retail_etl/run_id=manual__2024-06-06T18:26:48.106274+00:00/task_id=load_to_mongodb/attempt=1.log
[2024-06-06, 18:28:56 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: online_retail_etl.load_to_mongodb manual__2024-06-06T18:26:48.106274+00:00 [queued]>
[2024-06-06, 18:28:56 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: online_retail_etl.load_to_mongodb manual__2024-06-06T18:26:48.106274+00:00 [queued]>
[2024-06-06, 18:28:56 UTC] {taskinstance.py:1331} INFO - Starting attempt 1 of 2
[2024-06-06, 18:28:56 UTC] {taskinstance.py:1350} INFO - Executing <Task(PythonOperator): load_to_mongodb> on 2024-06-06 18:26:48.106274+00:00
[2024-06-06, 18:28:56 UTC] {standard_task_runner.py:57} INFO - Started process 9707 to run task
[2024-06-06, 18:28:56 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'online_retail_etl', 'load_to_mongodb', 'manual__2024-06-06T18:26:48.106274+00:00', '--job-id', '400', '--raw', '--subdir', 'DAGS_FOLD
[2024-06-06, 18:28:56 UTC] {standard_task_runner.py:85} INFO - Job 400: Subtask load_to_mongodb
[2024-06-06, 18:28:56 UTC] {task_command.py:410} INFO - Running <TaskInstance: online_retail_etl.load_to_mongodb manual__2024-06-06T18:26:48.106274+00:00 [running]> on host 0511406c3b60
[2024-06-06, 18:28:56 UTC] {taskinstance.py:1570} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='bralfarc7@alumnes.ub.edu' AIRFLOW_CTX_DAG_OWNER='Brandon and Eddie' AIRFLOW_CTX_DAG_ID='online_retail_etl' AIRFLOW_CTX_TASK_ID='lo
[2024-06-06, 18:28:56 UTC] {base.py:73} INFO - Using connection ID 'mongo_default' for task execution.
[2024-06-06, 18:28:56 UTC] {logger.py:96} INFO - {"message": "Waiting for suitable server to become available", "selector": "Primary()", "operation": "buildinfo", "topologyDescription": "<TopologyDescription id: 6661ffe8fb0f25b3eb
[2024-06-06, 18:28:58 UTC] {logging_mixin.py:149} INFO - Connected to MongoDB - {'version': '7.0.11', 'gitVersion': 'f451220f0df2b9dfe073f1521837f8ec5c208a8c', 'modules': ['enterprise'], 'allocator': 'tcmalloc', 'javascriptEngine'
[2024-06-06, 18:29:10 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 0: 10000
[2024-06-06, 18:29:20 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 10000: 10000
[2024-06-06, 18:29:29 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 20000: 10000
[2024-06-06, 18:29:39 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 30000: 10000
[2024-06-06, 18:30:25 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 40000: 10000
[2024-06-06, 18:31:13 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 50000: 10000
[2024-06-06, 18:31:28 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 60000: 10000
[2024-06-06, 18:31:37 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 70000: 10000
```



```
[2024-06-06, 18:35:33 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 240000: 10000
[2024-06-06, 18:35:43 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 250000: 10000
[2024-06-06, 18:36:02 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 260000: 10000
[2024-06-06, 18:36:12 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 270000: 10000
[2024-06-06, 18:36:21 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 280000: 10000
[2024-06-06, 18:36:31 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 290000: 10000
[2024-06-06, 18:36:40 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 300000: 10000
[2024-06-06, 18:37:09 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 310000: 10000
[2024-06-06, 18:37:49 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 320000: 10000
[2024-06-06, 18:38:21 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 330000: 10000
[2024-06-06, 18:38:31 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 340000: 10000
[2024-06-06, 18:38:40 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 350000: 10000
[2024-06-06, 18:39:06 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 360000: 10000
[2024-06-06, 18:39:46 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 370000: 10000
[2024-06-06, 18:40:15 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 380000: 10000
[2024-06-06, 18:40:25 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 390000: 10000
[2024-06-06, 18:40:35 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 400000: 10000
[2024-06-06, 18:40:44 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 410000: 10000
[2024-06-06, 18:41:38 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 420000: 10000
[2024-06-06, 18:42:11 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 430000: 10000
[2024-06-06, 18:42:22 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 440000: 10000
[2024-06-06, 18:42:31 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 450000: 10000
[2024-06-06, 18:42:41 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 460000: 10000
[2024-06-06, 18:43:08 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 470000: 10000
[2024-06-06, 18:43:18 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 480000: 10000
[2024-06-06, 18:43:24 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 490000: 10000
[2024-06-06, 18:43:34 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 500000: 10000
[2024-06-06, 18:43:43 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 510000: 10000
[2024-06-06, 18:44:19 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 520000: 10000
[2024-06-06, 18:44:47 UTC] {logging_mixin.py:149} INFO - Duplicated documents in chunk starting at row 530000: 6641
[2024-06-06, 18:44:47 UTC] {logging_mixin.py:149} INFO - Data insertion completed, new documents inserted 0, duplicated documents 536641
[2024-06-06, 18:44:47 UTC] {python.py:183} INFO - Done. Returned value was: 0
[2024-06-06, 18:44:47 UTC] {taskinstance.py:1373} INFO - Marking task as SUCCESS. dag_id=online_retail_etl, task_id=load_to_mongodb, execution_date=20240606T182648, start_date=20240606T182856, end_date=20240606T184447
[2024-06-06, 18:44:47 UTC] {local_task_job_runner.py:232} INFO - Task exited with return code 0
[2024-06-06, 18:44:47 UTC] {taskinstance.py:2674} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

Version: v2.6.0
Git Version: .release:ab54c63940a99646df974d4bcf2e37415e277e69

* Task 5: send_email_on_success



DAG: online_retail_etl ETL pipeline for Online Retail dataset

Task Instance: send_email_on_success at 2024-06-06, 18:15:23

Log by attempts

```
*** Found local files:
***   * /opt/airflow/logs/dag_id=online_retail_etl/run_id=manual__2024-06-06T18:15:23.754499+00:00/task_id=send_email_on_success/attempt=1.log
[2024-06-06, 18:26:05 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: online_retail_etl.send_email_on_success manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:26:05 UTC] {taskinstance.py:1125} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: online_retail_etl.send_email_on_success manual__2024-06-06T18:15:23.754499+00:00 [queued]>
[2024-06-06, 18:26:05 UTC] {taskinstance.py:1331} INFO - Starting attempt 1 of 2
[2024-06-06, 18:26:05 UTC] {taskinstance.py:1350} INFO - Executing <Task(EmailOperator): send_email_on_success> on 2024-06-06 18:15:23.754499+00:00
[2024-06-06, 18:26:05 UTC] {standard_task_runner.py:57} INFO - Started process 9641 to run task
[2024-06-06, 18:26:05 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'online_retail_etl', 'send_email_on_success', 'manual__2024-06-06T18:15:23.754499+00:00', '--job-id', '396', '--raw', '--subdir', 'DAG
[2024-06-06, 18:26:05 UTC] {standard_task_runner.py:85} INFO - Job 396: Subtask send_email_on_success
[2024-06-06, 18:26:05 UTC] {task_command.py:410} INFO - Running <TaskInstance: online_retail_etl.send_email_on_success manual__2024-06-06T18:15:23.754499+00:00 [running]> on host 0511406c3b60
[2024-06-06, 18:26:05 UTC] {taskinstance.py:1570} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='bralfarc7@alumnes.ub.edu' AIRFLOW_CTX_DAG_OWNER='Brandon and Eddie' AIRFLOW_CTX_DAG_ID='online_retail_etl' AIRFLOW_CTX_TASK_ID='se
[2024-06-06, 18:26:05 UTC] {warnings.py:110} WARNING - /home/***/.local/lib/python3.7/site-packages/***/utils/email.py:152: RemovedInAirflow3Warning: Fetching SMTP credentials from configuration variables will be deprecated in a f
  send_mime_email(e_from=mail_from, e_to=recipients, mime_msg=msg, conn_id=conn_id, dryrun=dryrun)
[2024-06-06, 18:26:05 UTC] {email.py:268} INFO - Email alerting: attempt 1
[2024-06-06, 18:26:06 UTC] {email.py:280} INFO - Sent an alert email to ['bralfarc7@alumnes.ub.edu']
[2024-06-06, 18:26:06 UTC] {taskinstance.py:1373} INFO - Marking task as SUCCESS. dag_id=online_retail_etl, task_id=send_email_on_success, execution_date=20240606T181523, start_date=20240606T182605, end_date=20240606T182606
[2024-06-06, 18:26:06 UTC] {local_task_job_runner.py:232} INFO - Task exited with return code 0
[2024-06-06, 18:26:06 UTC] {taskinstance.py:2674} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

Authors: Brandon Jersaí Alfaro Checa, Eddie Conti

Docker extras

In order to work with packages that are not included within the original Basic Airflow cluster configuration for CeleryExecutor with Redis and PostgreSQL, we need to use a custom image containing the necessary dependencies we use in our DAG tasks: 'pandas', 'pymongo', 'openpyxl' and MongoDB's Airflow provider: 'apache-airflow-providers-mongo'.

For this matter, we created a DockerFile adding the necessary packages on top of the base Apache Airflow 'apache/airflow:2.6.0':

```
FROM apache/airflow:2.6.0

RUN pip install pymongo apache-airflow-providers-mongo pandas requests
openpyxl
```

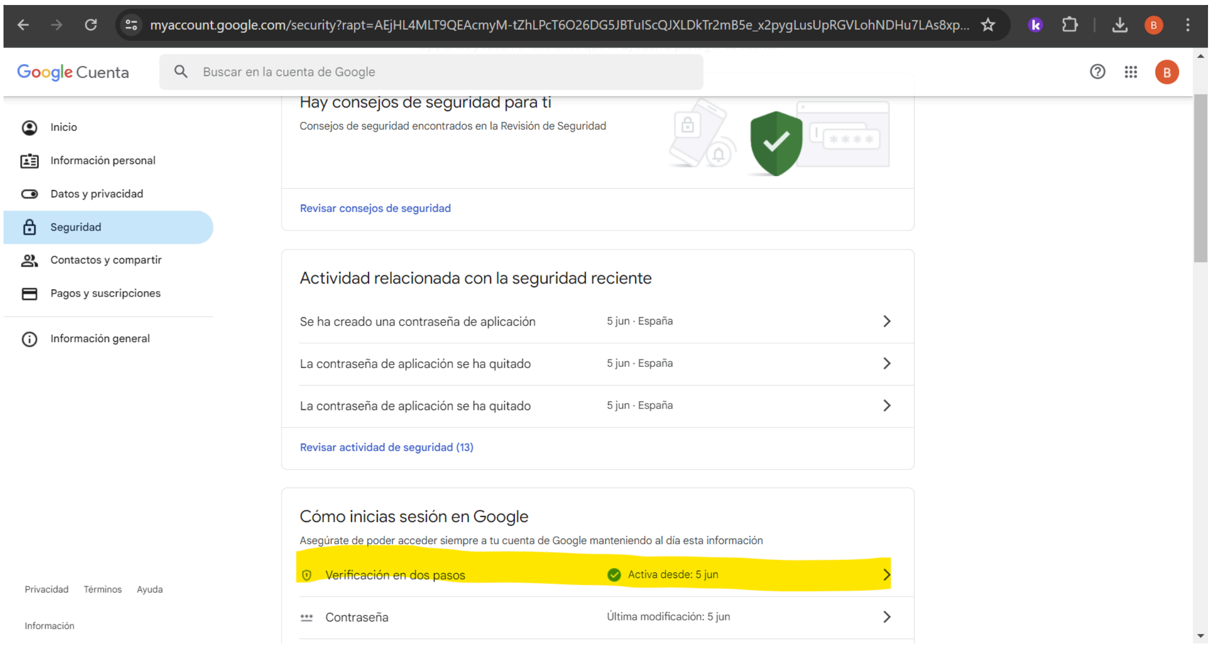
Then we can just add the extended image into our docker-compose.yml file to build and run the extended Airflow image with CeleryExecutor, Redis, and PostgreSQL.

```
version: '3.8'
x-airflow-common:
  &airflow-common
  # In order to add custom dependencies or upgrade provider packages you can
use your extended image.
  # Comment the image line, place your Dockerfile in the directory where you
placed the docker-compose.yaml
  # and uncomment the "build" line below, Then run `docker-compose build` to
build the images.
  image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:2.6.0}
  build: .
  environment:
    &airflow-common-env
    AIRFLOW__CORE__EXECUTOR: CeleryExecutor
    AIRFLOW__DATABASE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
    # For backward compatibility, with Airflow <2.3
    AIRFLOW__CORE__SQL_ALCHEMY_CONN:
postgresql+psycopg2://airflow:airflow@postgres/airflow
    AIRFLOW__CELERY__RESULT_BACKEND:
db+postgresql://airflow:airflow@postgres/airflow
    AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
    AIRFLOW__CORE__FERNET_KEY: ''
    AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: 'true'
    AIRFLOW__CORE__LOAD_EXAMPLES: 'true'
    AIRFLOW__API__AUTH_BACKENDS:
'airflow.api.auth.backend.basic_auth,airflow.api.auth.backend.session'
    AIRFLOW__SMTP__SMTP_HOST: smtp.gmail.com
    AIRFLOW__SMTP__SMTP_STARTTLS: 'true'
```

Authors: Brandon Jersaí Alfaro Checa, Eddie Conti

```
    AIRFLOW__SMTP__SMTP_SSL: 'false'
    AIRFLOW__SMTP__SMTP_USER: ${SMTP_USER}
    AIRFLOW__SMTP__SMTP_PASSWORD: ${SMTP_PASSWORD}
    AIRFLOW__SMTP__SMTP_PORT: '587'
    AIRFLOW__SMTP__SMTP_MAIL_FROM: ${SMTP_USER}
```

Note that we also added several environment variables, which refer to the Simple Mail Transfer Protocol (SMTP) configuration needed to send email notifications. For this DAG, we used a gmail.com email account to send the emails.
One important aspect to note here is that Gmail will not allow Airflow to connect into the email account, because of security reasons. For that matter, we first need to turn-on the two-step verification with your Gmail account:


myaccount.google.com/security?rapt=AEjHL4MLT9QEAcmyM-tZhLPcT6O26DG5JBTuIScQJXLDkTr2mB5e_x2pygLusUpRGVLohNDHu7LAs8xp...

Google Cuenta

Buscar en la cuenta de Google

Inicio
Información personal
Datos y privacidad
Seguridad
Contactos y compartir
Pagos y suscripciones
Información general

Privacidad Términos Ayuda
Información

Hay consejos de seguridad para ti
Consejos de seguridad encontrados en la Revisión de Seguridad
Revisar consejos de seguridad

Actividad relacionada con la seguridad reciente

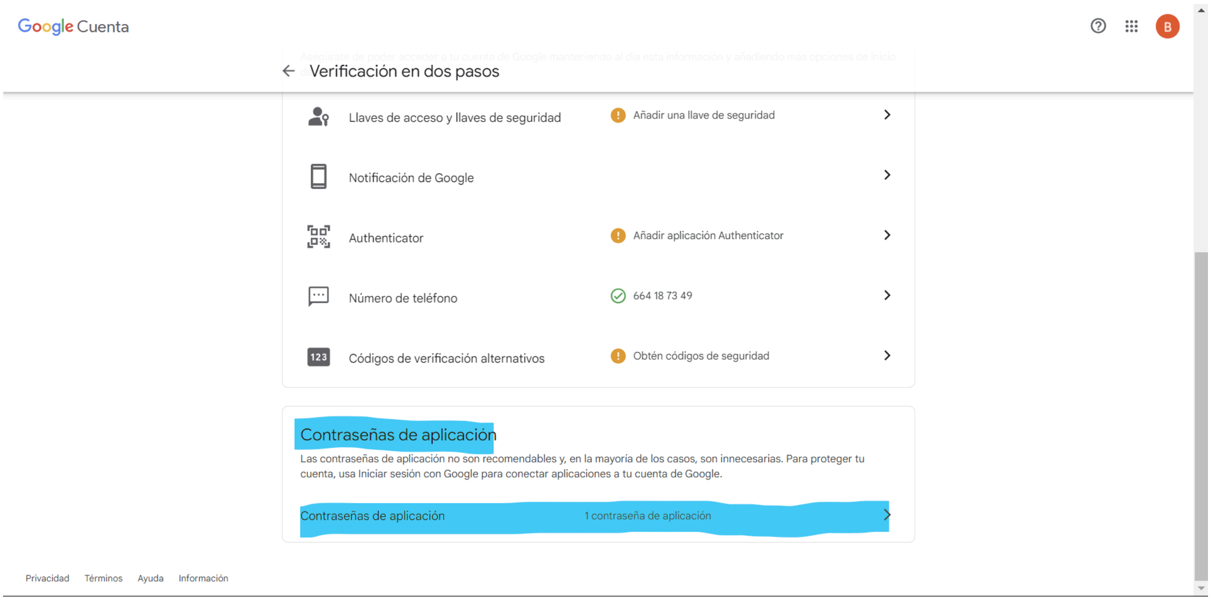
Se ha creado una contraseña de aplicación	5 jun · España
La contraseña de aplicación se ha quitado	5 jun · España
La contraseña de aplicación se ha quitado	5 jun · España

Revisar actividad de seguridad (13)

Cómo inicias sesión en Google
Asegúrate de poder acceder siempre a tu cuenta de Google manteniendo al día esta información

Verificación en dos pasos	Activa desde: 5 jun
Contraseña	Última modificación: 5 jun

Then, we would be able to generate an app password, which is a a 16-digit passcode that gives a less secure app or device permission to access your Google Account.


Google Cuenta
Verificación en dos pasos
Llaves de acceso y llaves de seguridad
Añadir una llave de seguridad
Notificación de Google
Authenticator
Añadir aplicación Authenticator
Número de teléfono
664 18 73 49
Códigos de verificación alternativos
Obtén códigos de seguridad

Contraseñas de aplicación

Las contraseñas de aplicación no son recomendables y, en la mayoría de los casos, son innecesarias. Para proteger tu cuenta, usa Iniciar sesión con Google para conectar aplicaciones a tu cuenta de Google.

Contraseñas de aplicación
1 contraseña de aplicación
Privacidad
Términos
Ayuda
Información

After generating this password, we can just copy it into our AIRFLOW__SMTP__SMTP_PASSWORD environment variable, and email notifications will be set successfully.