Airflow Operators need to Die

By Bolke de Bruin
Bolke de Bruin

Airflow since 2015

Now maintaining serialization and deserialization of Xcom
Why Operators?

➔ **Parallelism and Dependency Management**
  Defining dependencies between tasks in a workflow.

➔ **Modularity**
  Encapsulate a specific type of task or operation

➔ **Abstraction**
  Abstract away the details of executing specific operations

➔ **Extensibility**
  Organizations can create custom Operators to suit their needs
Apache Airflow boosts over **100 Operators** in > **80** provider packages
What are the most used Operators in Airflow?
PythonOperator & BashOperator.

(> 90% as reported by Astronomer, > 50% by Google)
So do we then spend a lot of effort on operators that are mostly unused?
But there are other issues...

You cannot use them *within* Taskflow functions or within PythonOperator

O(n):
- HiveToMySql
- MySqlToHive
- VerticaToHive
- GlacierToGcs

It's hard to capture lineage from Operators
Meet Marcos.

Marcos is a one-man-army data scientist and lives and breathes Notebooks.

He works for an auction company and deals with lots, hammer values, nas, gas etc.
if mlflow.active_run():
    display("Found a current active MLflow run. Ending the current one and restarting. You probably do not want this")
    mlflow.end_run()
mlflow.start_run(experiment_id=EXPERIMENT_ID)
mlflow.log_param("platform", PLATFORM_ID)

Logged 1 run to an experiment in MLflow. Learn more

# this grabs the intercom data and transforms it so we can use it
intercom_df = spark.read.format("delta").load(INTERCOM_IN)

# remove html from message subject and body
intercom_df = intercom_df.withColumn("subject", F.regexp_replace("subject", "html", "")
intercom_df = intercom_df.withColumn("body", F.regexp_replace("body", "html", ""))
Rethinking the Operator

Long live the Operator!
Wouldn’t it be nice if...
...Operators could deal with File-like objects residing anywhere?

So that GlacierToGCS (and basically every Transfer Operator) could just become:

CopyFileObject(src, dst)

Capturing lineage at the same time?
Taskflow and Operators would be truly integrated

So that this becomes possible

```python
@task
def my_task():
    for i in range(10):
        do_something()
    S3ToMySqlOperator(sql)
```

Instead of:

```python
@task
def my_task():
    hook = MySQLHook()
    hook.execute(sql)
```
…Operators know how to deal with DataFrames

```python
import(dataframe, snowflake_table)
df = export(snowflake_table)
```

Removing the need for *DatastoreToDatastore* Operators
This would reduce the number of Operators by ~70%

Splitting between Tasks and Data Aware Operators
The Astro SDK covers some of this, but not all.
Let’s embrace and extend

- BlobAPI
- DataFrameAPI
- Remove unneeded Operators
Thank you!

bdbruin@gmail.com