UDACITY

< Return to Classroom

DISCUSS ON STUDENT HUB

Data Pipelines with Airflow

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 specification requires changes

Good job on the project so far. Your work is almost perfect! The pipeline manages to correctly stage and update the tables.

There is one more update needed to pass all the specifications of this project:

- The DAG must contain all parameters required in the rubric.

If you need more help with this project, do not hesitate to ask questions in the Knowledge Hub.

Good luck with your next submission!

General

DAG can be browsed without issues in the Airflow UI

The dag follows the data flow provided in the instructions, all the tasks have a dependency and DAG

begins with a start_execution task and ends with a end_execution task.

Great work with the DAG. The dependencies graph is correct and all tasks have dependencies.

I see that you have used bitwise operators when drawing the dependencies e.g.

```
start_operator >> stage_events_to_redshift >> load_songplays_table
```

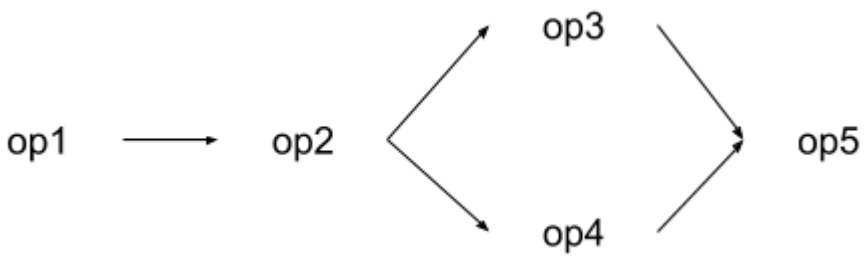
That is good! This is the recommended way of writing DAG dependencies as explained in the Airflow's concepts documentation:

> We recommend you setting operator relationships with bitshift operators rather than `set_upstream()` and `set_downstream()`.

To make diagram look even more concise, you may even combine them in a single line:

```
op1 >> op2 >> [op3, op4] >> op5
```

It will make a diagram that looks like the following:



Dag configuration

DAG contains default_args dict, with the following keys:

- **Owner**
- **Depends_on_past**
- **Start_date**
- **Retries**
- **Retry_delay**
- **Catchup**

As requested in the project instructions, your DAG default parameters must have the following values:

Configuring the DAG

In the DAG, add `default parameters` according to these guidelines

- The DAG does not have dependencies on past runs
- On failure, the task are retried 3 times
- Retries happen every 5 minutes
- Catchup is turned off
- Do not email on retry

Please revisit the values for the following arguments:

```
    'depends_on_past' : True,
    'retries' : 1,
    'retry_delay' : timedelta(minutes=1),
    'catchup' : True
```

The DAG object has default args set

The DAG should be scheduled to run once an hour

Good work with the DAG scheduling.

I see that you are using a shorthand `@hourly` in your DAG code. This is a great way to write the schedule interval as it is more readable. If you need to write more complicated rules, however, you need to use cron expressions.

You may learn the full extent of cron expressions in this CronTrigger Tutorial.

Check also this cool web tool to interpret Cron syntax: https://crontab.guru/

Staging the data

There is a task that to stages data from S3 to Redshift. (Runs a Redshift copy statement)

It is very nice that you have also included the REGION setting in the COPY command. As explained in the

documentation for COPY from Amazon S3, REGION setting is needed if the Amazon S3 buckets that hold the data files don't reside in the same AWS Region as your cluster, as is often the case for many ETL projects.

Instead of running a static SQL statement to stage the data, the task uses params to generate the copy statement dynamically

Good job setting the COPY statement to accept parameters.

The operator contains logging in different steps of the execution

Logging has been implemented, well done.

Note that when when logging information, it is better to use `log.info()` rather than `log.warning()`:

```
self.log.warning("Data quality check passed")
```

The SQL statements are executed by using a Airflow hook

Good job using Airflow hooks in your project to load the SQL statements.

Loading dimensions and facts

Dimensions are loaded with on the LoadDimension operator

Facts are loaded with on the LoadFact operator

Instead of running a static SQL statement to stage the data, the task uses params to generate the copy statement dynamically

The DAG allows to switch between append-only and delete-load functionality

The option to switch between append-only and delete-load functionality has been included through the parameter `is_delete_load` in your code.

Data Quality Checks

Data quality check is done with correct operator

Good job performing data quality checks. Quality checking may seem like a trivial task, but doing this would save you from headaches later on. In my case, I often thought I have transformed the data properly, only to find that some observations were missing due to errors in the ETL process. To work around this, I get into the habit of thinking about the number of observations before and after the ETL process and use the conclusion to create a quality check.

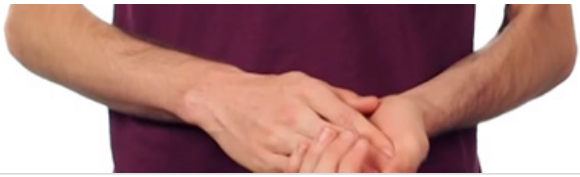
The DAG either fails or retries n times

Operator uses params to get the tests and the results, tests are not hard coded to the operator

In your project, you have made the data quality operator accept the `dq_checks` parameter that can be set with the statements to check and their expected results. Excellent work!

RESUBMIT

DOWNLOAD PROJECT



Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

START