Common Pitfalls

Overview

Airflow has a lot of great features and is a fast moving project. As such, there are some common pitfalls that are worth noting.

Details

- We do support more than one DAG definition per python file, but it is not recommended as we would like better isolation between DAGs from a fault and deployment perspective and multiple DAGs in the same file goes against that. For now, make sure that the dag object is in the global namespace : you can use the globals dict as in `globals()[dag_id] = DAG(...)`
- Configuring parallelism in airflow.cfg
    - parallelism = number of physical python processes the scheduler can run
    - dag_concurrency = the number of TIs to be allowed to run PER-dag at once
    - max_active_runs_per_dag = number of dag runs (per-DAG) to allow running at once
- Understanding the `execution date`
    - Airflow was developed as a solution for ETL needs. In the ETL world, you typically summarize data. So, if I want to summarize data for `2016-02-19`, I would do it at `2016-02-20 midnight GMT`, which would be right after all data for `2016-02-19` becomes available.
    - This date is available to you in both Jinja and a Python callable's context in many forms as documented here. As a note `ds` refers to `date_string`, not `date start` as may be confusing to some.
- Run your entire Airflow infrastructure in UTC. Airflow was developed at Airbnb, where every system runs on UTC (GMT). As a result, various parts of Airflow assume that the system (and database) timezone is UTC (GMT). This includes:
    - Webserver
    - Metadata DB
    - Scheduler
    - Workers (possibly)
- When setting a schedule, align the start date with the schedule. If a schedule is to run at 2am UTC, the start-date should also be at 2am UTC
- Bash Operator - Jinja templating and the bash commands
    - Described here : see below. You need to add a space after the script name in cases where you are directly calling a bash scripts in the `bash_command` attribute of`BashOperator` - this is because the Airflow tries to apply a Jinja template to it, which will fail.

```
t2 = BashOperator(
task_id='sleep',
bash_command="/home/batcher/test.sh", // This fails with `Jinja template not found` error
#bash_command="/home/batcher/test.sh ", // This works (has a space after)
dag=dag)
```

- When needing to change your start_date and schedule interval, change the name of the dag (a.k.a. dag_id) - I follow the convention : `my_dag_v1`, `my_dag_v2`, `my_dag_v3`, `my_dag_v4`, etc...
- When talking to MySQL, please use `mysqlclient`, not pymysql
- Using a start_date of datetime.now() can lead to unpredictable behavior, and your DAG never starting. See this post for details. It's recommended to subtract a timespan to force the scheduler to recognize the start_date. ***
- Solving MySQL Server has gone away
    - you may have an issue with your connection pool keeping connections open too long and checking out an old connection for you. To get a fresh connection, you can set the sql_alchemy_pool_recycle option in your configuration. Set this sql_alchemy_pool_recycle to the number of seconds of time in the connection pool between checkouts that you want a new connection to be created instead of an existing connection to be returned.
- If you intend to use extended ASCII or Unicode characters in Airflow, you have to provide a proper connection string to Airflow MySQL database, which defines charset explicitly, ex.:`sql_alchemy_conn = mysql://airflow@localhost:3306/airflow?charset=utf8` Otherwise you will experience exceptions thrown by WTForms templating and other Airflow modules:`UnicodeDecodeError: 'ascii' codec can't decode byte 0xae in position 506: ordinal not in range(128)`

Other useful Tips and Tricks:

- Airflow: Tips, Tricks, and Pitfalls @ Handy
- Best practices with Airflow (Max) nov 2015