What is a Workflow?

The final part

Definition

Now you are familiar with the core building blocks of Airflow, let's review the concepts we have seen:

- DAG: a description of the order in which work should take place
- Operator: a class that acts as a template for carrying out some work
- Task: an instance of an operator
- Task Instance: a specific run of a task characterized as the combination of a dag, a task and a point in time.

By combining DAGs and Operators to create TaskInstances, you can build complex workflows.

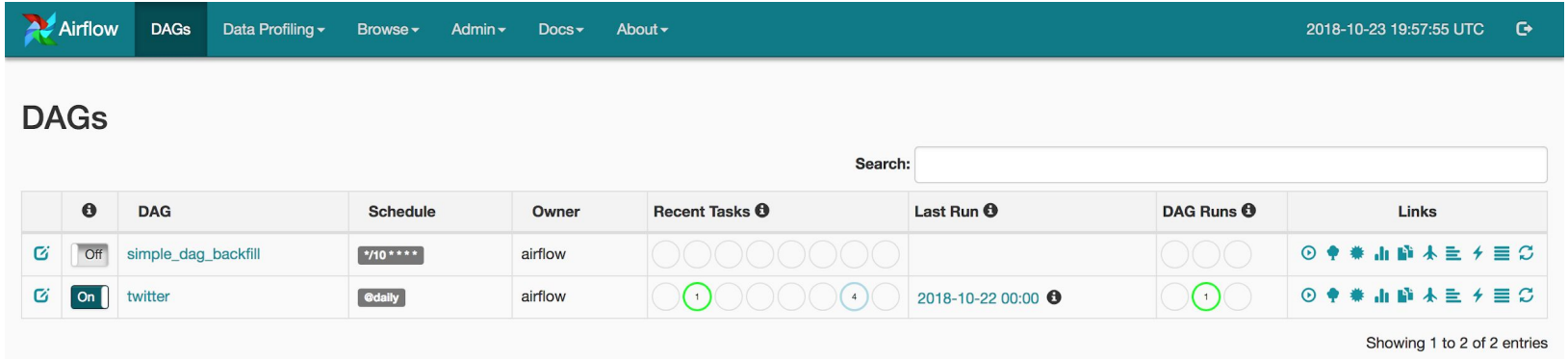
Tadaaaa!

You actually have done your first workflow! Congratulation !!

But wait.. Now it’s time to run it!

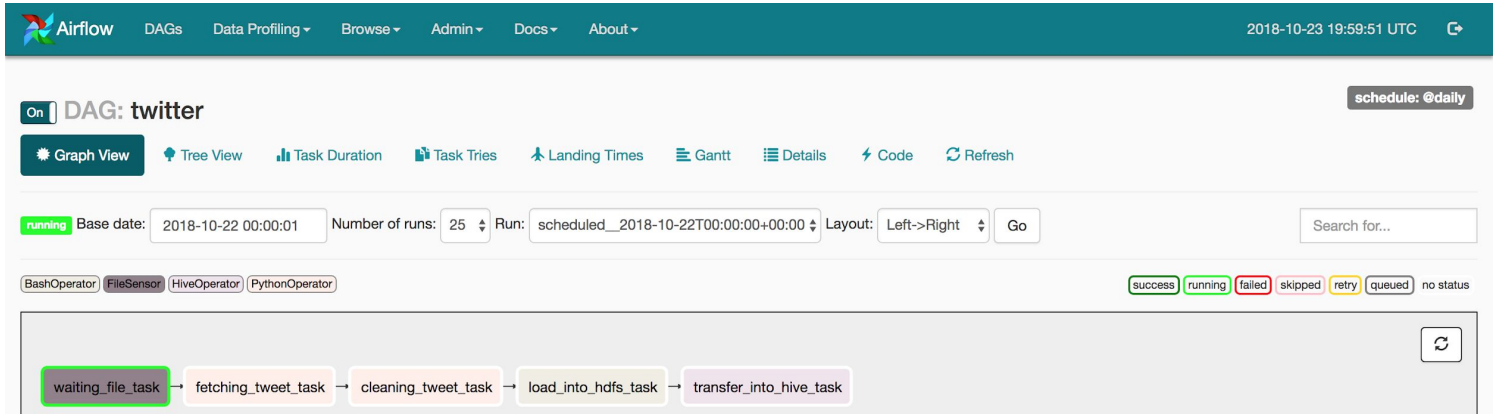
Instructions

- Turn ON the Scheduler toggle from the Airflow UI corresponding to twitter DAG. The DAG should run automatically.



Instructions

- You should see the waiting_file_task sensor running. It's normal since it loops until the data.csv file exists into /home/airflow/first_dag/



Instructions

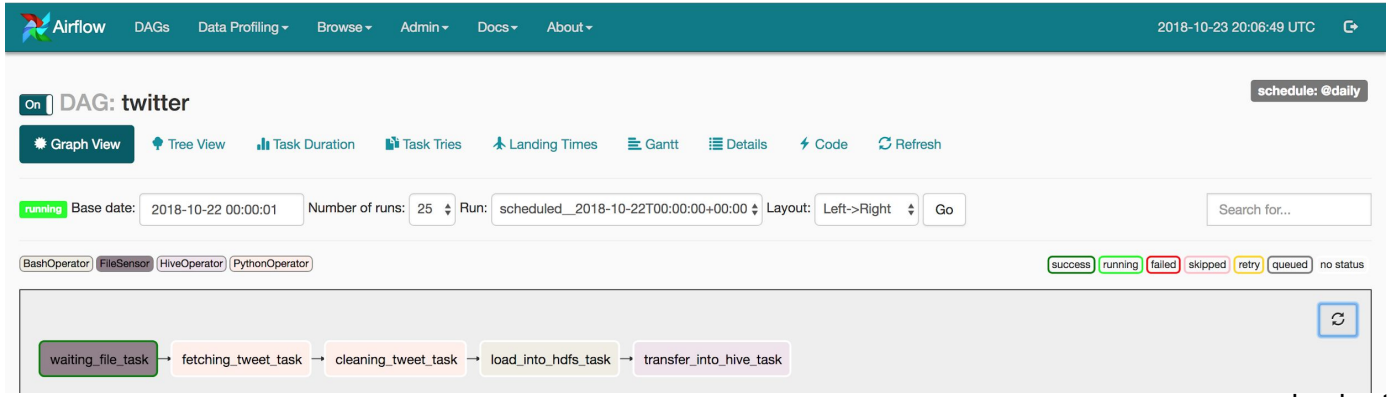
- If you want to see what is doing the sensor just click on the task from the “Graph View” then click on “View Log”. You should see the following lignes near the end of your logs:

```
[2018-10-23 19:59:13,164] {models.py:1569} INFO - Executing <Task(FileSensor): waiting_file_task> on 2018-10-22T00:00:00+00:00
[2018-10-23 19:59:13,164] {base_task_runner.py:124} INFO - Running: ['bash', '-c', 'airflow run twitter waiting_file_task 2018-10-22T00:00:00+00:00 --job_id 2 --raw -sd DAGS_FOLDE
[2018-10-23 19:59:13,752] {base_task_runner.py:107} INFO - Job 2: Subtask waiting_file_task [2018-10-23 19:59:13,745] {__init__.py:51} INFO - Using executor SequentialExecutor
[2018-10-23 19:59:13,856] {base_task_runner.py:107} INFO - Job 2: Subtask waiting_file_task [2018-10-23 19:59:13,847] {models.py:258} INFO - Filling up the DagBag from /home/airfl
[2018-10-23 19:59:14,131] {base_task_runner.py:107} INFO - Job 2: Subtask waiting_file_task [2018-10-23 19:59:14,127] {cli.py:492} INFO - Running <TaskInstance: twitter.waiting_fi
[2018-10-23 19:59:14,197] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 19:59:29,214] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 19:59:44,231] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 19:59:59,248] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
```

- Notice the “poke” every 15 seconds to check if data.csv exists.

Instructions

- Type:
  - `cp /home/airflow/first_dag/data/data.csv /home/airflow/first_dag/`
- Now if you refresh the “Graph View” by clicking on the little button at the top-right corner of the graph you should see waiting_file_task turning into dark green meaning the task has succeed.



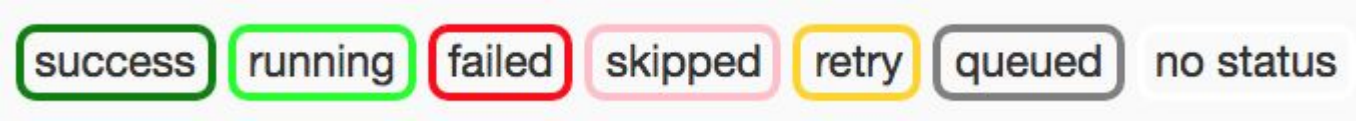
Instructions

- By the way, if you go back into the corresponding logs of waiting_file_task you can see the “poke” process finished because data.csv has been detected into /home/airflow/first_dag/

```
[2018-10-23 20:05:14,546] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 20:05:29,556] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 20:05:44,572] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 20:05:59,580] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 20:06:14,598] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 20:06:29,612] {file_sensor.py:60} INFO - Poking for file ///home/airflow/first_dag/data.csv
[2018-10-23 20:06:29,613] {base_sensor_operator.py:79} INFO - Success criteria met. Exiting.
[2018-10-23 20:06:33,751] {logging_mixin.py:95} INFO - [2018-10-23 20:06:33,751] {jobs.py:2612} INFO - Task exited with return code 0
```

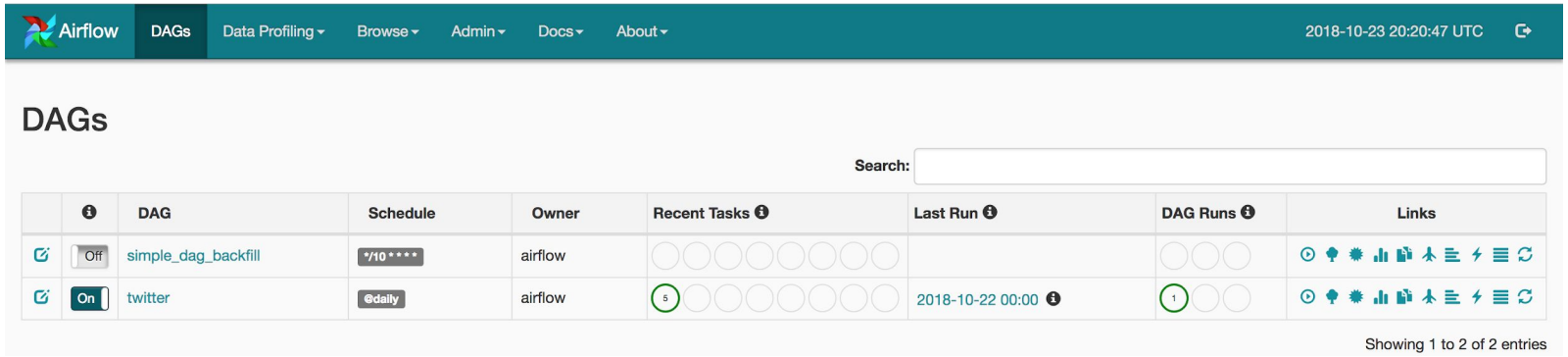
Instructions

- Finally, if you keep refreshing your DAG from the Graph View for instance, you will see your task turning into different colors corresponding to the different states in which a task/dag can be:



Instructions

- Once every task succeed you should see your DAG with 1 succeed DagRun and 5 succeed Tasks (since when a DAG is scheduled a DagRun is created the 5 tasks to run in it).



Instructions

- Now to check if everything is fine, you can go to hive by typing `hive` into your terminal.
- Wait until you see the prompt `hive>`
- Type: `SELECT COUNT(*) FROM tweets;`
- You should see the following result:

```
hive> SELECT COUNT(*) FROM tweets;
Query ID = airflow_20181023195247_26e22e5a-cea1-44da-8e8d-60bdd8ec68e7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-10-23 19:52:50,871 Stage-1 map = 100%,  reduce = 0%
2018-10-23 19:52:51,879 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1866066267_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1225492 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
3218
```

3128 tweets