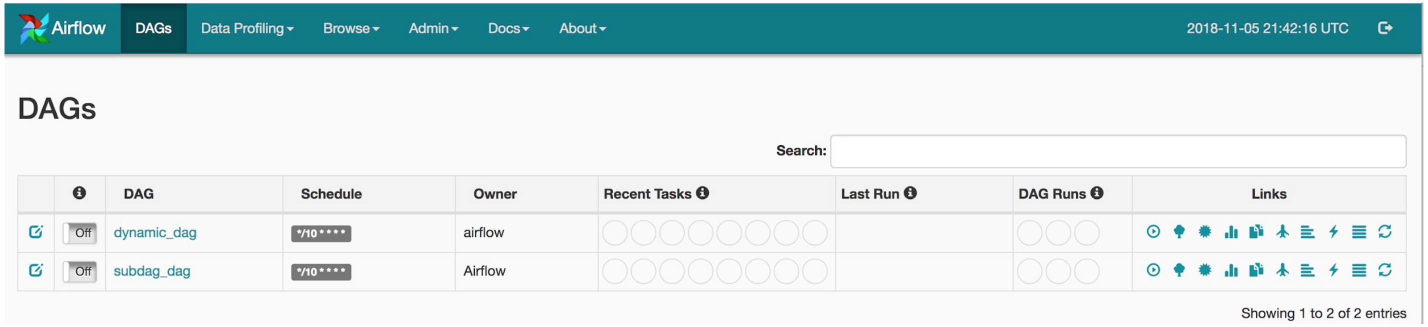
Minimising a DAG with SubDAGs

Let’s use SubDAGs

Instructions

- `cd ~`
- `cp ~/airflow_files/subdag_dag.py ~/airflow/dags/`
- If the airflow scheduler and webserver are not running, start them by typing:
  a. `airflow webserver`
  b. `airflow scheduler`
- After a bit of time you should see the DAG subdag_dag displayed as follow:



DAGs

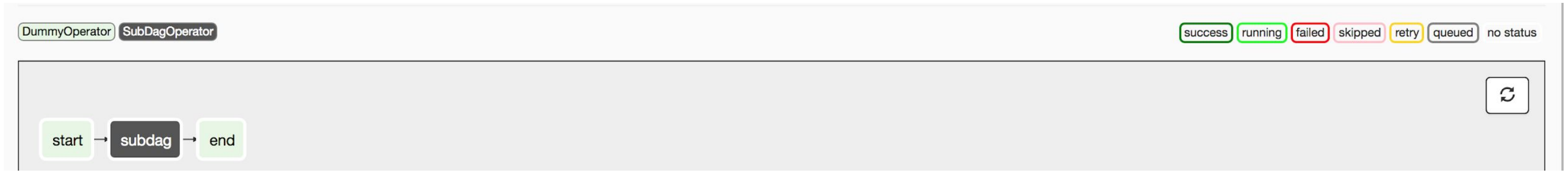
Search:

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	Off	dynamic_dag	*/10 * * * *	airflow				
	Off	subdag_dag	*/10 * * * *	Airflow				

Showing 1 to 2 of 2 entries

Instructions

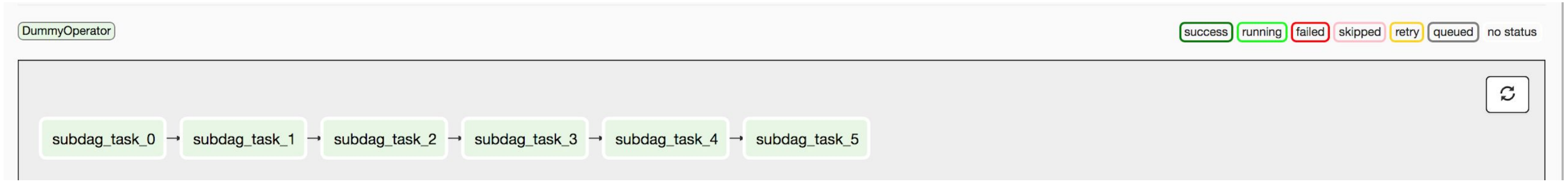
- Click on “subdag_dag” and “Graph View” you should have the following DAG:



- Notice the colored task in dark grey meaning this task is a SubDAG. Also, don’t hesitate to refer to the legend of operators just above giving you information about which task corresponds to which Operator.
- Without the factory function we would have ended up with another DAG listed in Airflow UI.

Instructions

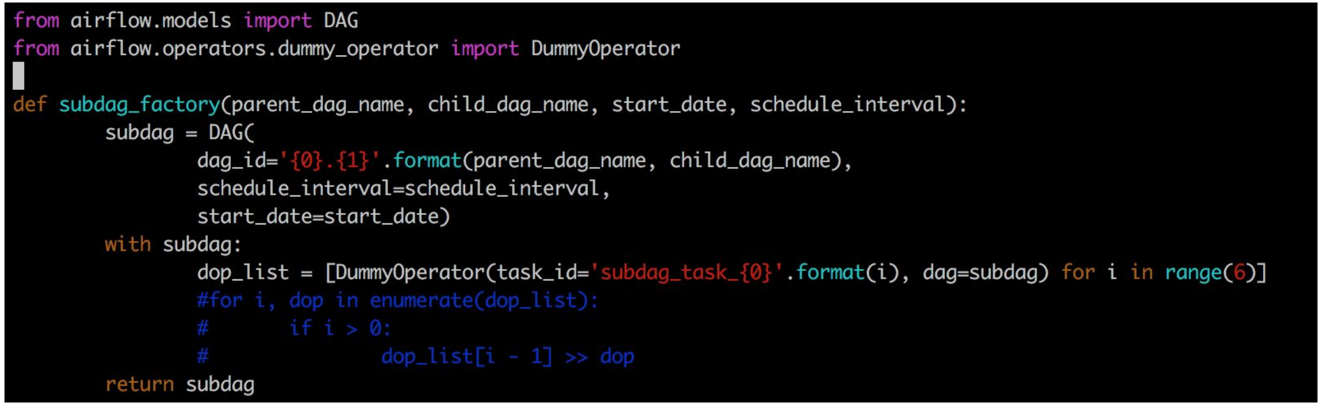
- Click on the “subdag” task.
- In the dialog box, click on “Zoom into Sub DAG”
- Then, by clicking on “Graph View”, you should see that we are effectively looking at the graph of the subdag we created.



- It’s exactly like we have a child DAG into a parent DAG with its own dependencies. Here, task_0 will be first executed, then task_1 and so on, sequentially

Instructions

- Now, click on “back to subdag_dag”.
- We are going to comment the line of code making dependencies between the SubDAG’s tasks.
- `vim ~/airflow_files/subdag_factory.py`
- You should end up with the same code (Notice the # to comment lines):

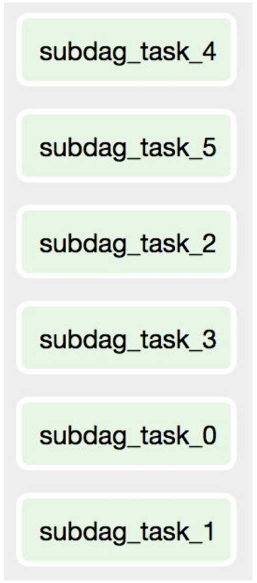


```
from airflow.models import DAG
from airflow.operators.dummy_operator import DummyOperator

def subdag_factory(parent_dag_name, child_dag_name, start_date, schedule_interval):
        subdag = DAG(
                dag_id='{0}.{1}'.format(parent_dag_name, child_dag_name),
                schedule_interval=schedule_interval,
                start_date=start_date)
        with subdag:
                dop_list = [DummyOperator(task_id='subdag_task_{0}'.format(i), dag=subdag) for i in range(6)]
                #for i, dop in enumerate(dop_list):
                #       if i > 0:
                #               dop_list[i - 1] >> dop
        return subdag
```

Instructions

- Now, if you go back to “subdag_dag” then “Graph View” and click on “subdag” task to “Zoom into Sub DAG”.
- From the “Graph View” of the SubDAG you should have now the following DAG:
- As expected there is no dependencies between the tasks of the SubDAG anymore.
- Important: You could think that by using Local Executor with dag_concurrency > 1 you would end up with tasks running in parallel but they won’t. Why? Because SubDagOperator uses its own executor passed as parameter. By default, Airflow sets this parameter to Sequential Executor in order to avoid possible bad performances. See more here



Important Notes

- By convention, a SubDAG's dag_id should be prefixed by its parent and a dot.
- Share arguments between the main DAG and the SubDAG by passing arguments to the SubDagOperator.
- SubDAGs must have a schedule and be enable as their parent. Same start_date and same schedule_intervale.
- Clearing a SubDagOperator clears the state of the tasks within.
- It's clearly advised to stay with Sequential Executor if you want to use SubDags. Using Local Executor or Celery Executor could produce unexpected behavior from your DAG.