

Apache Airflow in Open Data Hub

Vedant Mahabaleshwarkar
Red Hat AI Center of
Excellence

Introduction to Airflow

- Airflow is a platform created by community to programmatically author, schedule and monitor workflows.
- Airflow has a python SDK that can be used to define DAGs.
- Airflow has a large community, as well as some commercial offerings.
- Airflow can be run on bare-metal as well as on Kubernetes or OpenShift clusters.

Airflow Python SDK sample

```
<snip>
dag = DAG(
    dag_id='example_python_operator', default_args=args,
    schedule_interval=None)
<snip>
run_this = PythonOperator(
    task_id='print_the_context',
    provide_context=True,
    python_callable=print_context,
    dag=dag)

for i in range(10):
    '''
    Generating 10 sleeping task, sleeping from 0 to 9 seconds
    respectively
    '''
    task = PythonOperator(
        task_id='sleep_for_'+str(i),
        python_callable=my_sleeping_function,
        op_kwargs={'random_base': float(i)/10},
        dag=dag)

task.set_upstream(run_this)
```

Features in Apache Airflow

- **Repeatability:** Airflow has the ability to schedule workflows to run repeatedly like a cron job.
- **Failure alerting:** Airflow has some nice features like email alerts that can be defined for workflow failures.
- Ability to **process historical data:** Airflow allows backfill for DAGs and also stores information on the history of the DAGs run.
- Parallel processing and **dependency management:** Tasks in the workflows can be set to have upstream dependencies, tasks with no dependencies can execute in parallel.

Features in Apache Airflow

- Airflow has a huge selection of Operators that make interacting with other systems easier.
 - DAGs determine how to run a workflow, operators are (usually) a atomic task in the workflow (eg: bash operator, email operator)
- Airflow has executor options like CeleryExecutor, LocalExecutor, and KubernetesExecutor.
- KubernetesExecutor is a nice feature for cloud deployments because it enables Airflow to scale out, the executor creates and deletes worker pods to execute and manage tasks for a given DAG

Airflow UI

DAGs

Search:

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
<input type="button" value="On"/>	example_bash_operator	0 0 ***	Airflow				
<input type="button" value="On"/>	example_gcs_to_sftp	None	airflow				
<input type="button" value="On"/>	example_trigger_controller_dag	@once	airflow				
<input type="button" value="On"/>	example_trigger_target_dag	None	Airflow				
<input type="button" value="On"/>	test_backfill_pooled_task_dag	1 day, 0:00:00	airflow				
<input type="button" value="On"/>	test_default_impersonation	1 day, 0:00:00	airflow				
<input type="button" value="On"/>	test_task_view_type_check	1 day, 0:00:00	airflow				

Showing 1 to 7 of 7 entries

Hide Paused DAGs

Airflow UI

The screenshot displays the Apache Airflow web interface. At the top, there is a navigation bar with the Airflow logo and menu items: DAGs, Data Profiling, Browse, Admin, Docs, and About. The current date and time are shown as 2018-09-07 22:15:40 UTC. The main content area shows a DAG titled 'DAG: example_branch_dop_operator_v3' with a status of 'On' and a schedule of '*/1 * * * *'. Below the title, there are several view options: Graph View, Tree View (selected), Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Refresh, and Delete. A search bar for the DAG is present. Below the search bar, there are input fields for 'Base date: 2018-09-05 01:04:00' and 'Number of runs: 25', with a 'Go' button. A legend identifies the operators: BranchPythonOperator and DummyOperator. A legend for task statuses is also provided: success (dark green), running (light green), failed (red), skipped (pink), retry (yellow), queued (grey), and no status (white). The Gantt chart shows a timeline from 05:45 to 06 PM. The DAG structure is shown on the left, with nodes: [DAG], oper_1, condition, oper_2, and condition. The Gantt chart shows a sequence of tasks for each operator, with most tasks in a 'success' state (dark green).

Airflow integration with Open Data Hub

- Open Data Hub will deploy the Airflow Operator to manage Airflow as an application.
- We are using the Airflow Operator originally developed in the GoogleCloudPlatform repository and later donated to Apache.
- The Operator creates a controller-manager pod which will be created as a part of the Open Data Hub deployment.
- Users can then install the Airflow components they need from the available options (eg: CeleryExecutor or KubernetesExecutor, Postgres deployment or MySQL deployment etc.)

DEMO