Creating a Hello World DAG

Assuming that Airflow is already setup, we will create our first *hello world* DAG. All it will do is print a message to the log.

Below is the code for the DAG.

```
from datetime import datetime
from airflow import DAG
from airflow.operators.dummy_operator import DummyOperator
from airflow.operators.python_operator import PythonOperator

def print_hello():
    return 'Hello world from first Airflow DAG!'

dag = DAG('hello_world', description='Hello World DAG',
          schedule_interval='0h 12 * * *',
          start_date=datetime(2017, 3, 20), catchup=False)

hello_operator = PythonOperator(task_id='hello_task',
python_callable=print_hello, dag=dag)

hello_operator
```

We place this code (DAG) in our **AIRFLOW_HOME** directory under the **dags** folder. We name it *hello_world.py*.

```
├── airflow-webserver-monitor.pid
├── airflow-webserver.err
├── airflow-webserver.log
├── airflow-webserver.out
├── airflow-webserver.pid
├── airflow.cfg
├── airflow.db
├── airflow.db-journal
├── dags
│   ├── __pycache__
│   │   └── hello_world.cpython-36.pyc
│   └── hello_world.py
│
```

Let us understand what we have done in the file:

- In the first few lines, we are simply importing a few packages from **airflow**.
- Next, we define a function that prints the hello message.
- After that, we declare the DAG. It takes arguments such as **name**, **description**, **schedule_interval**, **start_date** and **catchup**. Setting catchup to false prevents Airflow from having the DAG runs catch up to the current date.
- Next, we define the operator and call it the **hello_operator**. In essence, this uses the in-built **PythonOperator** to call our **print_hello** function. We also provide a task_id to this operator.
- The last statement specifies the order of the operators. In this case, we have only one operator.

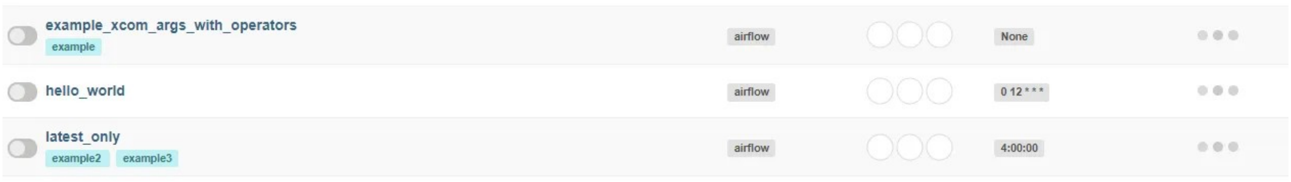
Running the DAG

To run the DAG, we need to start the **Airflow scheduler** by executing the below command:

```
airflow scheduler
```

Airflow scheduler is the entity that actually executes the DAGs. By default, we use **SequentialExecutor** which executes tasks one by one. In case of more complex workflow, we can use other executors such as **LocalExecutor** or **CeleryExecutor**.

If we have the **Airflow webserver** also running, we would be able to see our **hello_world** DAG in the list of available DAGs.

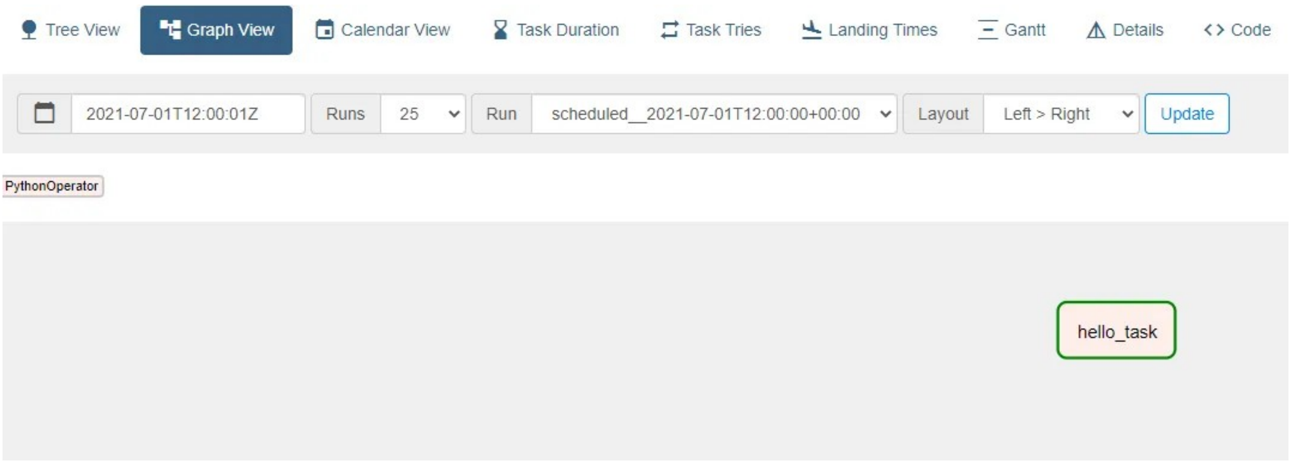


example_xcom_args_with_operators example	airflow	None	
hello_world	airflow	0 12 * * *	
latest_only example2 example3	airflow	4:00:00	

To start the DAG, we can to turn on the DAG by clicking the toggle button before the name of the DAG. As soon as that is done, we would be able to see messages in the scheduler logs about the DAG execution.

```
[2021-07-03 11:49:16,962] {scheduler_job.py:1212} INFO - Executor reports
execution of hello_world.hello_task execution_date=2021-07-01 12:00:00+00:00
exited with status success for try_number 1
[2021-07-03 11:49:17,100] {dagrun.py:444} INFO - Marking run <DagRun hello_world
@ 2021-07-01 12:00:00+00:00: scheduled__2021-07-01T12:00:00+00:00, externally
triggered: False> successful
```

We can also see the DAG graph view where the **hello_world** operator has executed successfully.



2021-07-01T12:00:01Z | Runs 25 | Run scheduled__2021-07-01T12:00:00+00:00 | Layout Left > Right

By clicking on the task box and opening the logs, we can see the logs as below:

```
[2021-07-03 11:49:16,755] {python.py:151} INFO - Done. Returned value was: Hello
world from first Airflow DAG!
```

```
[2021-07-03 11:49:16,768] {taskinstance.py:1191} INFO - Marking task as SUCCESS.
dag_id=hello_world, task_id=hello_task, execution_date=20210701T120000,
start_date=20210703T061916, end_date=20210703T061916
[2021-07-03 11:49:16,781] {taskinstance.py:1245} INFO - 0 downstream tasks
scheduled from follow-on schedule check
[2021-07-03 11:49:16,820] {local_task_job.py:151} INFO - Task exited with return
code 0
```

Here, we can see the hello world message. In other words, our DAG executed successfully and the task was marked as SUCCESS.

With this **Airflow DAG Example**, we have successfully created our first DAG and executed it using Airflow. Though it was a simple hello message, it has helped us understand the concepts behind a DAG execution in detail.