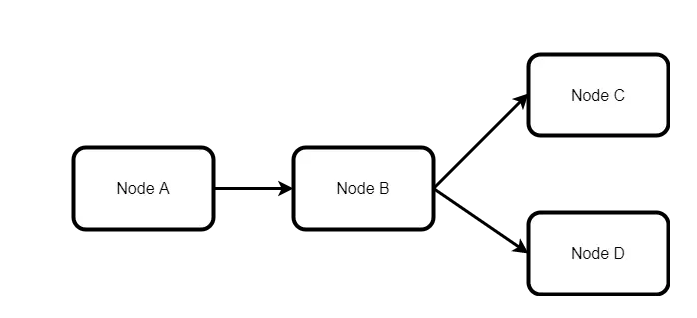
Notes on AirFlow

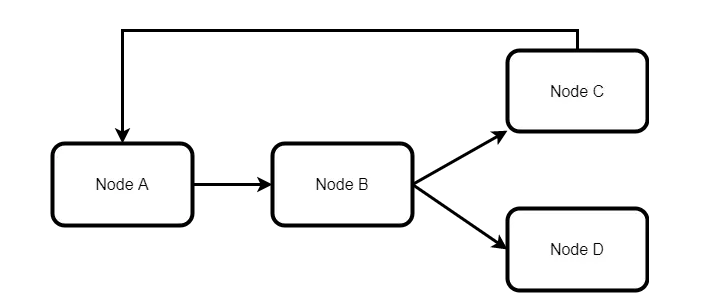
1 – What is a DAG?

A **workflow**in Airflow is designed as a **Directed Acyclic Graph (DAG)**. In other words, while designing a workflow, we should think of dividing the workflow into small tasks that can execute independently of each other.

In simple terms, a **DAG is a graph with nodes connected via directed edges**. Also, there should be no cycles within such a graph. For example, the below diagram represents a DAG.



However, the below example is not a **DAG**:



Ref: <https://progressivecoder.com/airflow-dag-example-create-your-first-dag/>

It is because there is a cycle in the second diagram from Node C to Node A. Due to this cycle, this DAG will not execute. However, the first diagram is a valid DAG.

A valid **DAG**can execute in an **Airflow installation**. Whenever, a DAG is triggered, a DAGRun is created. We can think of a **DAGrun**as an instance of the DAG with an execution timestamp.

## 2 – What are Nodes in a DAG?

The next aspect to understand is the meaning of a Node in a DAG. A Node is nothing but an operator.

To elaborate, an operator is a class that contains the logic of what we want to achieve in the DAG. For example, if we want to execute a Python script, we will have a **Python operator**. If we wish to execute a Bash command, we have **Bash operator**. There are several in-built operators available to us as part of Airflow. Documentation about them can be found [here](https://airflow.apache.org/docs/apache-airflow/stable/_api/airflow/operators/index.html).

When a particular operator is triggered, it becomes a task and executes as part of the overall DAG run.

Ref: <https://progressivecoder.com/airflow-dag-example-create-your-first-dag/>

# Hello World! using Apache-Airflow

Ref: <https://medium.com/international-school-of-ai-data-science/hello-world-using-apache-airflow-792947431455>

# Creating a Python File

* Enable your virtual environment and navigate to your airflow directory containing the dags folder and some other files.
* Open your favorite editor and create a new file with the name “**hello\_world\_dag.py**”.
* Place this code (DAG), i.e., **hello\_world\_dag.py,** in our AIRFLOW\_HOME directory under the **dags** folder.

# Importing the modules

* To create a proper pipeline in airflow, we need to **import**the “**DAG**” module and a python operator from the “**operators.python**” module in the airflow package.
* We will also import the “**datetime**” module to schedule the dags.

**from airflow import DAG  
from airflow.operators.python import PythonOperator  
from datetime import datetime**

# Creating a DAG object

* Next, we will instantiate a DAG object to nest the tasks in the pipeline. We pass on a “**dag\_id**” string which is the unique identifier of the dag.
* We recommend keeping the **python file**and **dag\_id** with the **same**nameand once done**,**we will assign the “**dag\_id**” as “**hello\_world\_dag**”.
* We will also set a “**start\_date**” parameter which indicates the timestamp from which the scheduler will attempt to backfill.
* It is followed by a “**schedule\_interval**” parameter which indicates the interval of subsequent DAG Runs created by the scheduler. It is in the form of a “**datetime.timedelta**” object or a cron expression. Airflow has a few cron presets available such as ‘**@hourly**’, ‘**@daily**’, ‘**@yearly**’, etc. You can read more about them [**here**](https://airflow.apache.org/docs/apache-airflow/stable/dag-run.html).
* If the “**start\_date**” is set as **January 1, 2021**, with a “**schedule\_interval**” of **hourly**, then the scheduler will start a DAG Run on an hourly basis until the present hour or the “**end\_date**” (optional parameter) has been reached. It is called **catchup,**and we can turn it off by keeping its parameter value as **False**.
* After setting these parameters, our DAG initialization should look like this:

**with DAG(dag\_id="hello\_world\_dag",  
 start\_date=datetime(2021,1,1),  
 schedule\_interval="@hourly",  
 catchup=False) as dag:**

# Creating a Task

* According to the airflow documentation, an object **instantiated**from an **operator**is called a **task**. In airflow, we have various types of operators, but for now, we will only focus on the **PythonOperator**.
* A PythonOperator is used to **call a python function** inside your DAG. We will create a PythonOperator object that calls a python function which will return “**Hello World”** upon its call.
* Like a DAG object has “**dag\_id”,** a PythonOperator object has an identifier called “**task\_id**”**.**
* It also has the “**python\_callable”** parameter, which takes the name of the callable function as its input.
* After setting the parameters, our task should look like this:

**task1 = PythonOperator(  
 task\_id="hello\_world",  
 python\_callable=helloWorld)**

# Creating a Callable Function

* We also need to create a function that will be called by the **PythonOperator**as shown below:

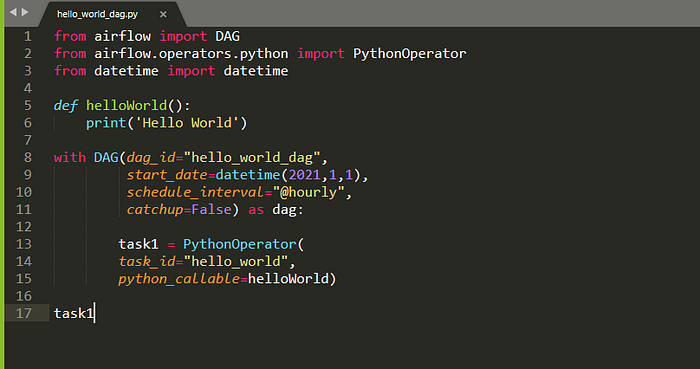
**def helloWorld():  
 print(‘Hello World’)**

# Setting Dependencies

* We can set the task dependencies by writing the **task names** along with **>>** or **<<** to indicate the **downstream** or **upstream**flow.
* Since we have a single task here, we don’t need to indicate the flow and proceed with writing the task name.

# Voila, it’s a DAG file

After compiling all the elements of the DAG, our **final code** should look like this:



A DAG file

We place **hello\_world\_dag.py,** in our AIRFLOW\_HOME directory under the **dags** folder.

├── airflow.cfg

├── airflow.db

├── dags

│   ├── \_\_pycache\_\_

│   │   └── hello\_world\_dag.cpython-36.pyc

│   └── hello\_world\_dag.py

## 4 – Running the DAG

To see the file running, activate the virtual environment and start your airflow webserver and scheduler.

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**.

We need to start the **Airflow webserver** by executing the below command in another terminal:

airflow webserver

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

* In Webbrowser UI, the DAG should run successfully. You can check the **graph view** or **tree view**by hovering over Links and selecting Graph or Tree options.
* You can also view the **task’s execution information** using **logs**. To do so, **click on the task,** which will lead you to the following dialog box: