[Apache-Airflow](https://airflow.apache.org/" \t "_blank) A practical guide

<https://medium.com/@srivathsankr7/apache-airflow-a-practical-guide-5164ff19d18b>

“Data really powers everything that we do.” — Jeff Weiner

In Artificial Intelligence era, Big Data has become the source to solve problems. Gone are those days where data is collected and processed in batches. Real-time data processing is the key for modern business.

Besides Volume Velocity Variety, managing data pipeline has a significant role to play when it comes to availability. The complex part is developing multiple data pipelines to address different use-cases and managing all at one place.

This article explains the usage of an open source platform which can programmatically author, schedule and monitor workflows for data pipelines.

*Yes, it is* [***Apache-Airflow***](https://airflow.apache.org/)*developed by* [***Airbnb Engineering***](https://airbnb.io/)

*“We chose it because we deal with huge amounts of data. Besides, it sounds really cool.” — Larry Page*

**Contents**

* Introduction
* Insight
* Setup
* Development
* Deployment
* Common shortfalls
* Conclusion

**Introduction**

Airflow framework can be used to build workflows. A workflow could be anything from a simple linux command to a complex hive queries, a python script to a Docker file. Workflow comprises one or more tasks which are connected by [Directed Acyclic Graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph). Workflow which is called as DAG in Airflow, can be executed manually or it can be automated with schedulers like [cron](https://en.wikipedia.org/wiki/Cron). Success and failure of DAG can be monitored, controlled and re-triggered. DAG state can be alerted with SMTP, [Slack](https://medium.com/u/26d90a99f605?source=post_page-----5164ff19d18b----------------------) and other systems.

*“With data collection, ‘the sooner the better’ is always the best answer.” —* Marissa Mayer

**Insight**

Airflow is developed in Python. It has following concepts

[**DAG**](https://airflow.apache.org/concepts.html#dags)**:** Connects independent tasks and executes in specified sequence.  
[**Task**](https://airflow.apache.org/concepts.html#tasks)**:** Logical unit of code.  
[**Operator**](https://airflow.apache.org/concepts.html#operators)**:** Template to wrap and execute task. BashOperator is used to execute bash script, PythonOperator is used to execute python code.

“The world is one big data problem. There’s a bit of arrogance in that, and a bit of truth as well.”–Andrew McAfee

**Setup**

This section explains how to install airflow in MacOS *High Sierra*

**[Installation - Airflow Documentation](https://airflow.apache.org/installation.html?source=post_page-----5164ff19d18b----------------------)**

**[One of the dependencies of Apache Airflow by default pulls in a GPL library ('unidecode'). In case this is a concern…](https://airflow.apache.org/installation.html?source=post_page-----5164ff19d18b----------------------)**

**[airflow.apache.org](https://airflow.apache.org/installation.html?source=post_page-----5164ff19d18b----------------------)**

**Install Python**

Download and install python from official website:   
<https://www.python.org/downloads/release/python-370/>  
I am using python 3.6  
To test if the installation is successful

python -V  
Python 3.6.X

Now install Airflow and its dependencies using pip

pip install apache-airflow==1.9.0  
pip install apache-airflow[celery]  
pip install mysqlclient

**Install RabbitMQ**

RabbitMQ website recommends installation via homebrew  
<https://www.rabbitmq.com/install-homebrew.html>

Default username and password for rabbitmq server is **guest**

brew update  
brew install rabbitmq

Add bin path to environment

$ vi .bash\_profile  
PATH=$PATH:/usr/local/sbin

To start Rabbitmq server in background

rabbitmq-server -detached

To check the status of Rabbitmq server

rabbitmqctl status

**Install MySQL**

Download MySQL for mac(.dmg) here  
<https://dev.mysql.com/downloads/mysql/>  
During installation process, set password for root user (I have selected legacy week password policy)  
After installation follow these steps to create airflow database

mysql -u root -p  
mysql> CREATE DATABASE airflow CHARACTER SET utf8 COLLATE utf8\_unicode\_ci;  
mysql> create user 'airflow'@'localhost' identified by 'airflow';  
mysql> grant all privileges on \* . \* to 'airflow'@'localhost';  
mysql> flush privileges;

**Initialize Airflow**

First we have to initialize airflow with the below command

airflow initdb

This creates airflow directory in home path with cfg files and logs folder. Create *dags* folder in airflow directory.

mkdir -p ~/airflow/dags/

We have to set these configurations before running our first airflow code

vi ~/airflow/airflow.cfg  
executor = CeleryExecutor  
sql\_alchemy\_conn = mysql://airflow:airflow@localhost:3306/airflow  
broker\_url = amqp://guest:guest@localhost:5672/

Here *SequentialExecutor* is default executor which can execute one DAG at a time. We use *CeleryExecutor* to execute multiple DAGs in parallel.  
Default database is *sqlite* which is not scalable. So we use mysql  
*CeleryExecutor* requires RabbitMQ serverwhich is configured in broker url

Now initialize airflow again to use MySQL as primary DB

airflow initdb

This creates necessary tables in MySQL *airflow* database

Now start airflow, worker and scheduler

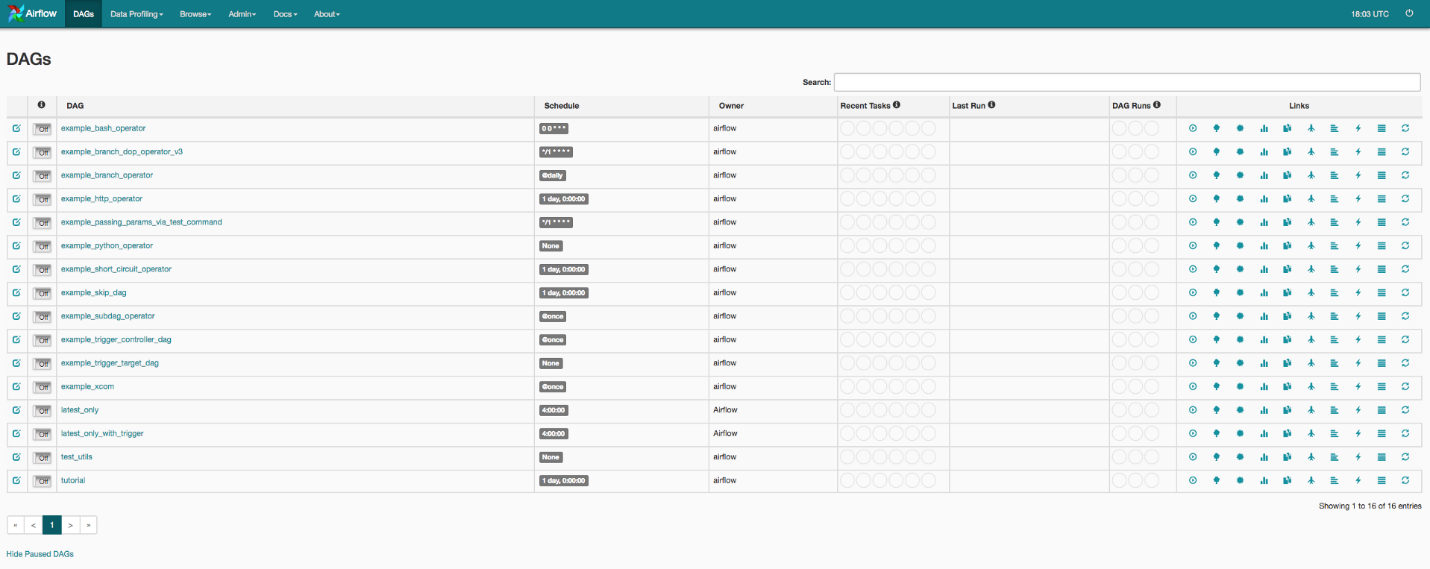
airflow webserver  
airflow worker  
airflow scheduler

Hit the URL

<http://localhost:8080/>

***Congrats!!*** Airflow is up and running 😆

https://miro.medium.com/max/60/1*PoVjVVuAlTEwaiTM2AO2Xg.png?q=20

<img class="cp t u lc ak" src="https://miro.medium.com/max/5112/1\*PoVjVVuAlTEwaiTM2AO2Xg.png" width="2556" height="1018" role="presentation"/>

Airflow home page

*“Data is the new science. Big Data holds the answers.” —* Pat Gelsinger

**Development**

Its time to get our hands dirty. We will write our first DAG python code using PythonOperator

from airflow.operators import PythonOperator  
from airflow.models import DAG  
from datetime import datetime, timedeltaargs = {'owner':'srivathsan', 'start\_date' : datetime(2018, 10, 1), 'retries': 2, 'retry\_delay': timedelta(minutes=1) }dags = DAG('test\_dag', default\_args = args)def print\_context(val):  
 print(val)def print\_text():  
 print('Hello-World')t1 = PythonOperator(task\_id='multitask1',op\_kwargs={'val':{'a':1, 'b':2}}, python\_callable=print\_context, dag = dags)t2 = PythonOperator(task\_id='multitask2', python\_callable=print\_text, dag=dags)t2.set\_upstream(t1)

Let us walk through line by line

***args*** is a dict of key value pairs passed as input to DAG

***dags*** is an instance of DAG with dag name ‘My\_first\_dag’ and args as parameter

***print\_context()*** and ***print\_text()*** are two python functions executed by DAG task

***t1*** and ***t2*** are PythonOperators with ***task\_id*** as task name, ***op\_kwargs*** as arguments passed to function and ***python\_callable***

***t2.set\_upstream(t1)*** considers t1 and t2 as vertices and connects as directed graph

**Deployment**

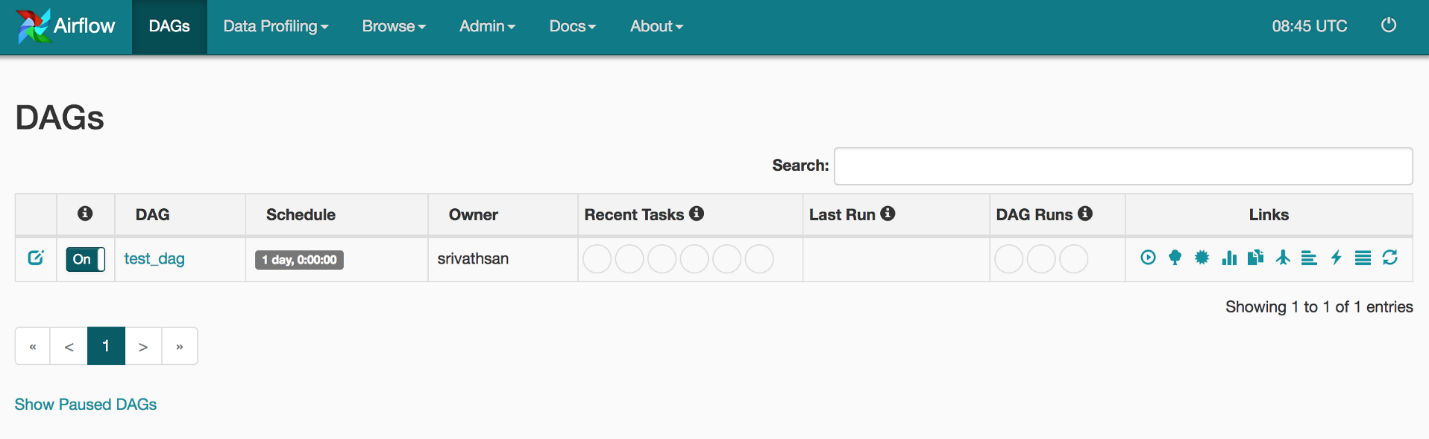
Deploying Airflow is the easiest task

cp airflow\_test.py ~/airflow/dags/

Copy the airflow python file and paste in ***airflow/dags/*** folder

Airflow will recognise changes in dags folder and populates new DAG in webUI

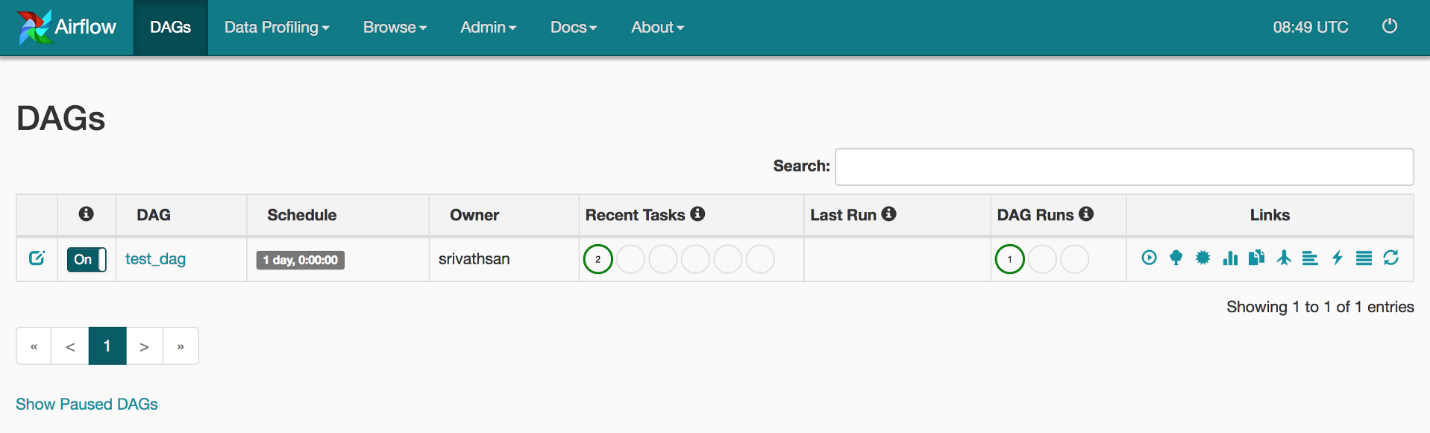
https://miro.medium.com/max/60/1*tdQ5O9ufo7yeDE8kNkHDUQ.png?q=20

<img class="cp t u lc ak" src="https://miro.medium.com/max/5108/1\*tdQ5O9ufo7yeDE8kNkHDUQ.png" width="2554" height="782" role="presentation"/>

test\_dag in Airflow UI

To execute test\_dag click ***play(>)*** button in ***Links*** column

https://miro.medium.com/max/60/1*PMR2-p9bOItl3riCCRNcBg.png?q=20

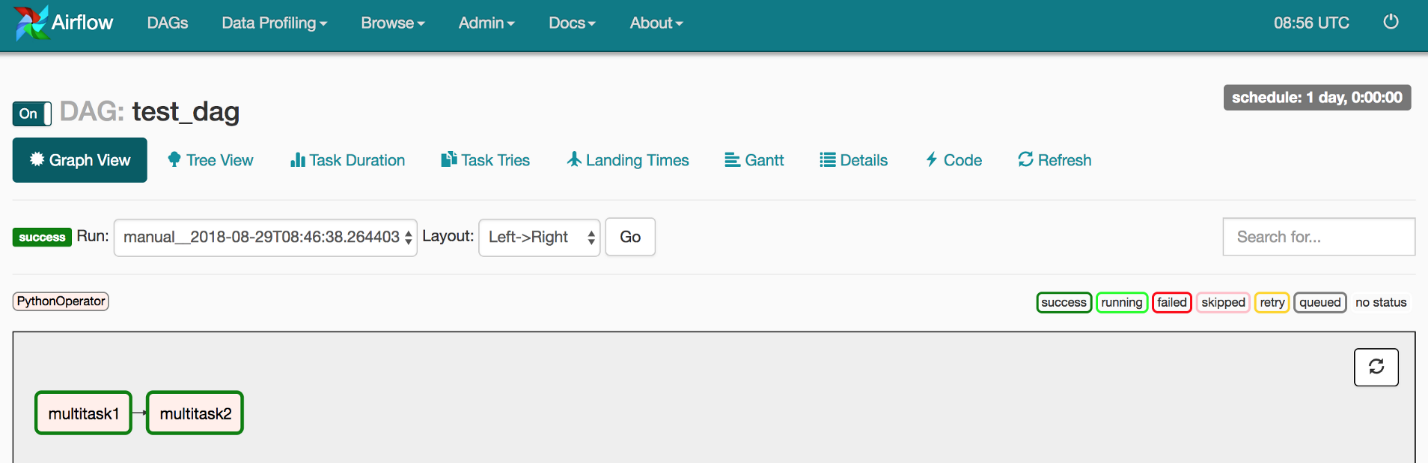
<img class="cp t u lc ak" src="https://miro.medium.com/max/5112/1\*PMR2-p9bOItl3riCCRNcBg.png" width="2556" height="774" role="presentation"/>

test\_dag after execution

After DAG execution, UI shows number of times the DAG executed in ***DAG Runs*** column and count of tasks executed in ***Recent Tasks*** column circled in green color indicating success.

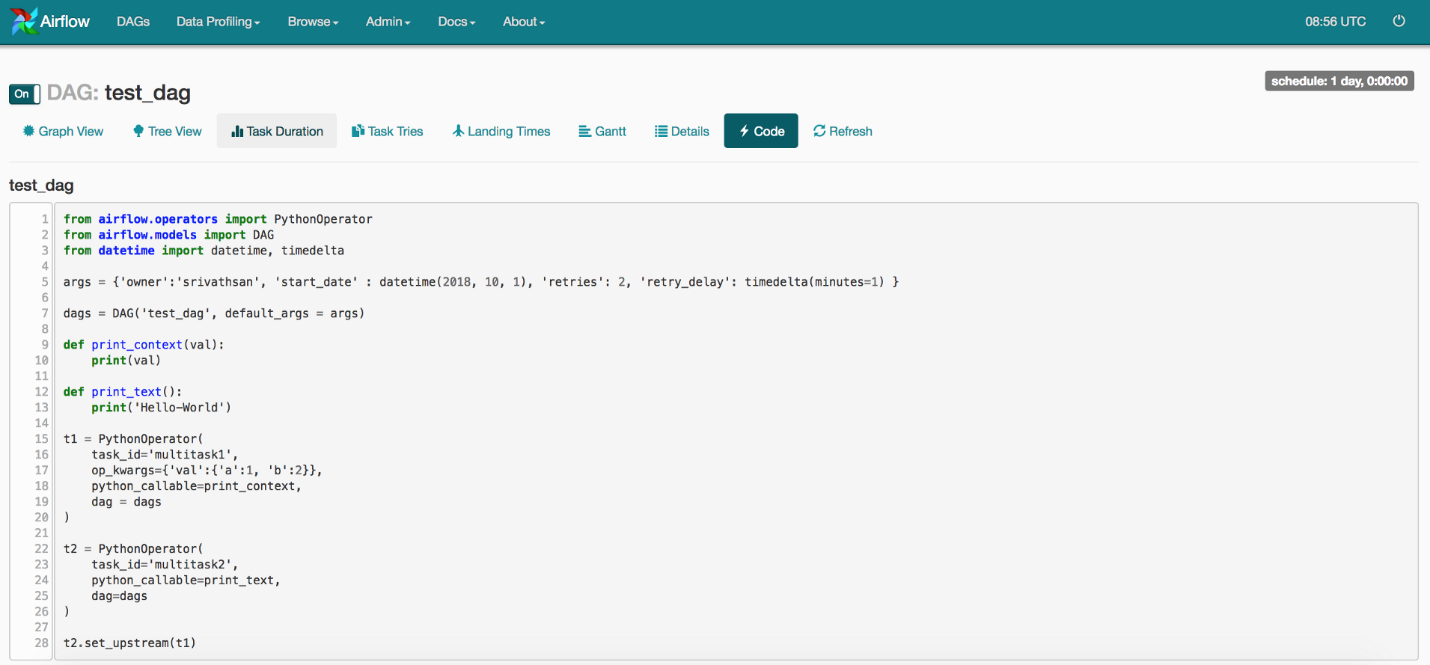
We can view task dependency graph and code by selecting DAG name in ***DAG*** column

https://miro.medium.com/max/60/1*YcpU_9fGAoe0Y0LHQHZs1Q.png?q=20

<img class="cp t u lc ak" src="https://miro.medium.com/max/5088/1\*YcpU\_9fGAoe0Y0LHQHZs1Q.png" width="2544" height="822" role="presentation"/>

test\_dag Graph View shows two tasks

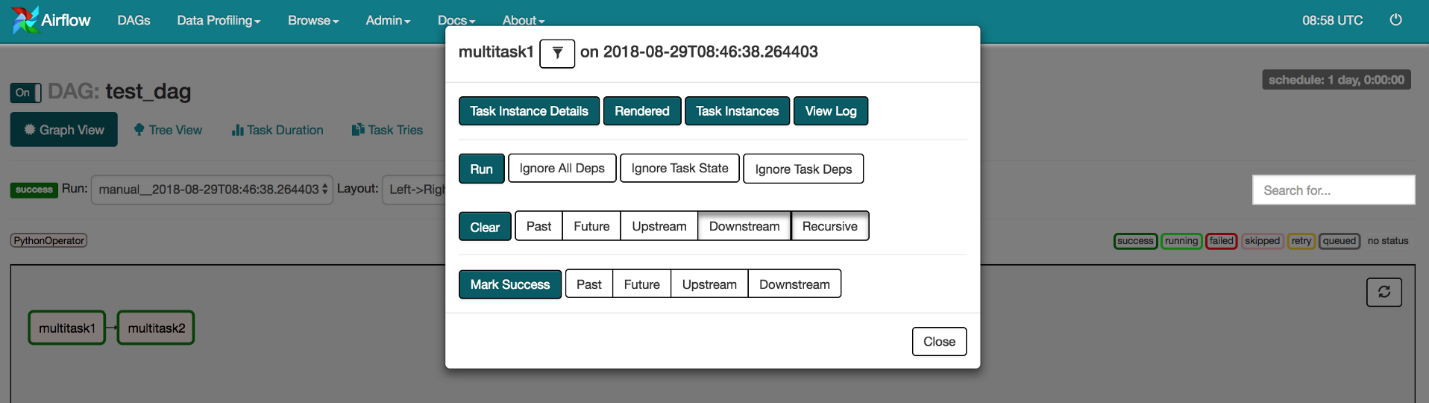
https://miro.medium.com/max/60/1*I-jCNvp2pgQiOLbw3x_DkQ.png?q=20

<img class="cp t u lc ak" src="https://miro.medium.com/max/5096/1\*I-jCNvp2pgQiOLbw3x\_DkQ.png" width="2548" height="1186" role="presentation"/>

code of test\_dag

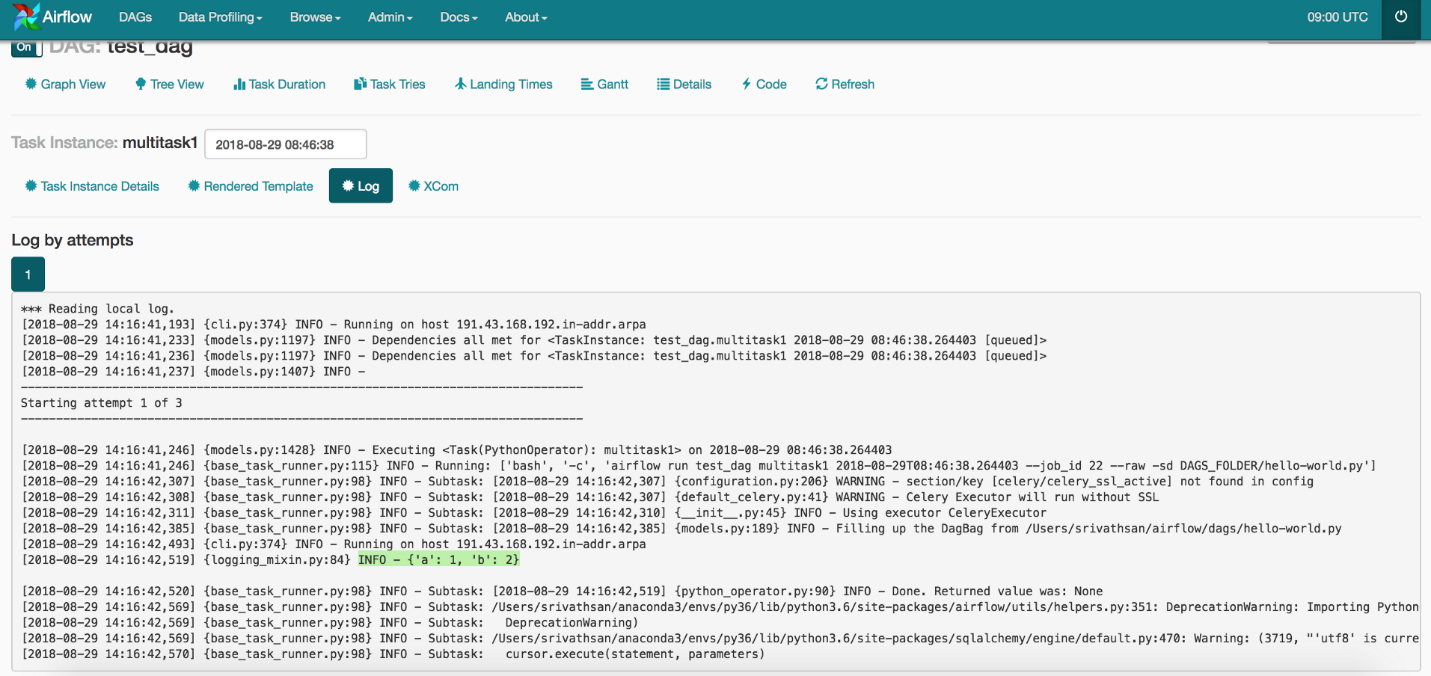
We can see the output of test\_dag tasks by selecting ***task name -> View Log***

https://miro.medium.com/max/60/1*wTvrbz2F8eVf-gNXOiGwHA.png?q=20

<img class="cp t u lc ak" src="https://miro.medium.com/max/5108/1\*wTvrbz2F8eVf-gNXOiGwHA.png" width="2554" height="720" role="presentation"/>

Selecting multitask1

https://miro.medium.com/max/60/1*TRf6iTbmLnTCgkhs6BaglQ.png?q=20

<img class="cp t u lc ak" src="https://miro.medium.com/max/5100/1\*TRf6iTbmLnTCgkhs6BaglQ.png" width="2550" height="1206" role="presentation"/>

Selecting View Log shows output of multitask1 highlighted in log

Now we have seen basic example of Airflow. There are many features to explore. Some of the notable features are

* **XComs** — Cross communication between DAGs
* **Authentication and Authorisation** — User management to login and access DAGs
* **Schedulers** — Schedule DAG to execute like cron job
* **SMTP , Slack**— To alert with automated email for success and failures
* **Retries and Retry delay** — Automate retry in case of failure and delay time between two retry attempts

***“There were 5 Exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days.” —*** Eric Schmidt

**Common shortfalls**

Most common pitfalls worth noticing before start

* pip install apache-airflow==1.9.0  
  Notice its ***apache-airflow***. Package name used to be airflow till 1.8 and renamed to apache-airflow after releasing to Apache community
* args = {‘owner’:’srivathsan’, ‘start\_date’ : **datetime(2018, 10, 1)**, ‘retries’: 2, ‘retry\_delay’: timedelta(minutes=1) }  
  These are few arguments passed to DAG() while creating dag instance. Avoid using ***datetime.now()*** because airflow scheduler requires only past datetime. Not present datetime.
* Make sure to start ***airflow webserver, airflow worker and airflow scheduler*** before executing a DAG. You can also put these services in background.  
  nohup airflow webserver >> ~/airflow/logs/webserver.log &  
  nohup airflow worker >> ~/airflow/logs/worker.log &  
  nohup airflow scheduler $\* >> ~/airflow/logs/scheduler.log &

“Everything is going to be connected to cloud and data… All of this will be mediated by software.” — Satya Nadella

**Conclusion**

simple and scalable platform with scheduling, monitoring and fault-tolerant capabilities in one word is ***Apache-Airflow.*** Credit goes to [**Airbnb**](https://www.airbnb.co.in/)for open source contribution and [AirbnbEng](https://medium.com/u/ebe93072cafd?source=post_page-----5164ff19d18b----------------------) team for designing and architecting fabulous framework.

Apache-Airflow — <https://airflow.apache.org/>  
Airflow github — <https://github.com/apache/incubator-airflow>  
Airbnb Engineering — <https://airbnb.io/>