What is Apache Airflow?

- Apache Airflow is a **workflow engine** that will **easily schedule** and **run** your **complex data pipelines**.
- It will make sure that **each task** of your **data pipeline** will get **executed** in the **correct order** and **each task** gets the **required resources**.

Features of Apache Airflow:

1) **Easy to Use:** If you have a bit of python knowledge, you are good to go and deploy on Airflow.
2) **Open Source:** It is free and open-source with a lot of active users.
3) **Robust Integrations:** It will give you ready to use operators so that you can work with Google Cloud Platform, Amazon Web Services, Microsoft Azure, etc.
4) **Use Standard Python to code:** You can use python to create simple to complex workflows with complete flexibility.
5) **Amazing User Interface:** You can monitor and manage your workflows. It will allow you to check the status of completed and ongoing tasks.

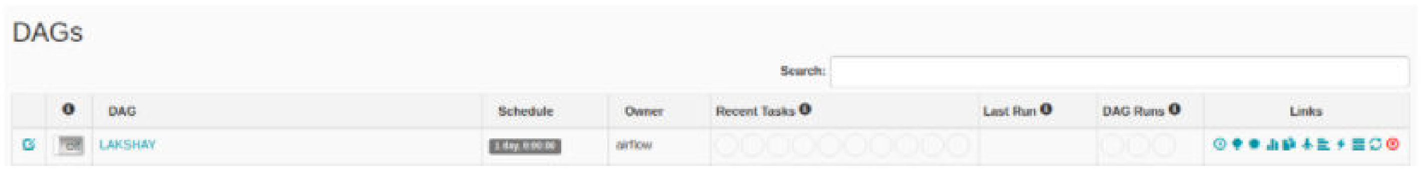
Components of Apache Airflow:

- **DAG:** It is the Directed Acyclic Graph – a **collection** of **all** the **tasks** that you want to **run** which is **organized** and **shows** the **relationship** between **different tasks**. It is defined in a python script.
- **Web Server:** It allows us to **monitor** the **status** of **DAGs** and **trigger** them. It is **user interface** built on **Flask**.
- **Metadata Database:** Airflow **stores** the **status** of **all tasks** in a **database** and do all **read/write operations** of a **workflow** from here.
- **Scheduler:** As the name suggests, this **component** is **responsible** for **scheduling** the **execution** of **DAGs**. It **retrieves** and **updates** the **status** of the **task** in the **database**.

User Interface:

1) DAG VIEW:

a) It is the **default view** of the **user interface**.
b) This will **list down all** the **DAGS present** in **your system**.
c) It will **give you** a **summarized view** of the **DAGS** like **how many times** a particular DAG was **run successfully**, how many times it **failed**, the **last execution time**, and some other useful links.



DAGs

Search:

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
		LAKSHAY	1 day, 0:00:00	airflow				

2) GRAPH VIEW:

a) In **graph** view, you can **visualize each** and **every step** of **your workflow** with their **dependencies** and their **current status**.

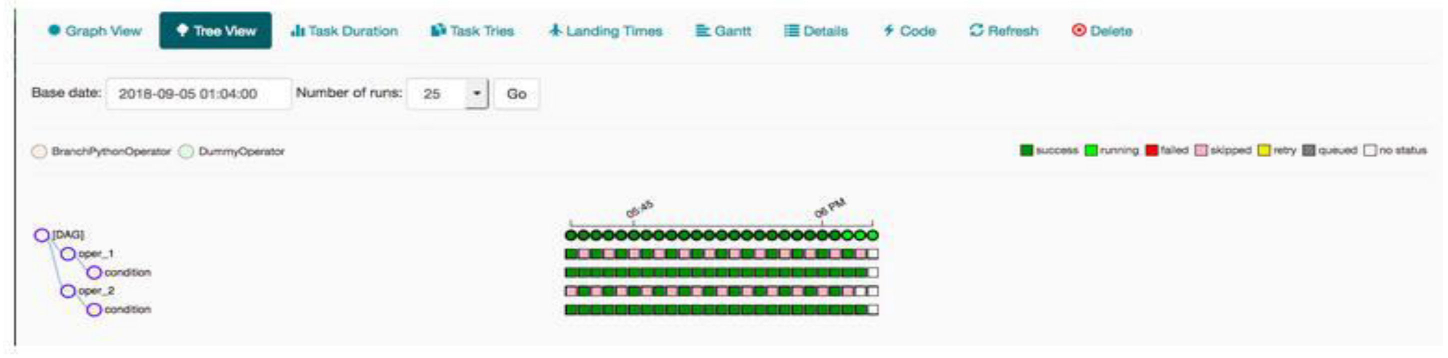
b) You can **check** the **current status** with **different color codes**.

🟩	Task is successfully completed.
🟩	Task is in progress.
🟥	Task failed
🟪	Task has been skipped
🟨	Task failed once, executor is retrying
🟦	Task is in the queue.
⬜	No Status

3) TREE VIEW:

a) The **tree view** also represents the **DAG**.

b) If you think your **pipeline took** a **longer time** to **execute** than **expected** then you can **check** which **part** is **taking** a **long time** to **execute** and **then you can work** on it.

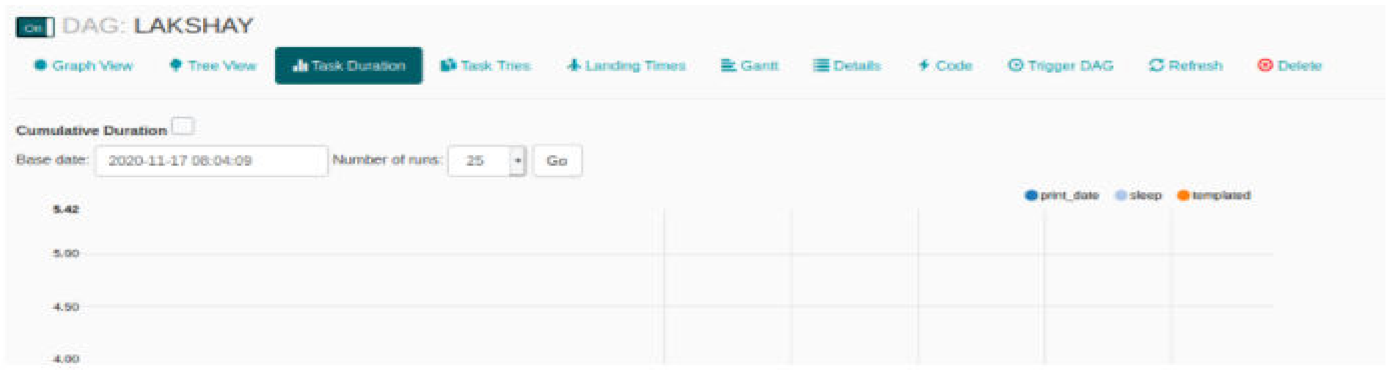


Base date: 2018-09-05 01:04:00 Number of runs: 25 Go

4) TASK DURATION:

a) In this view, you can **compare** the **duration** of your **tasks run** at **different time intervals**.

b) You can **optimize** your **algorithms** and **compare** your **performance** here.

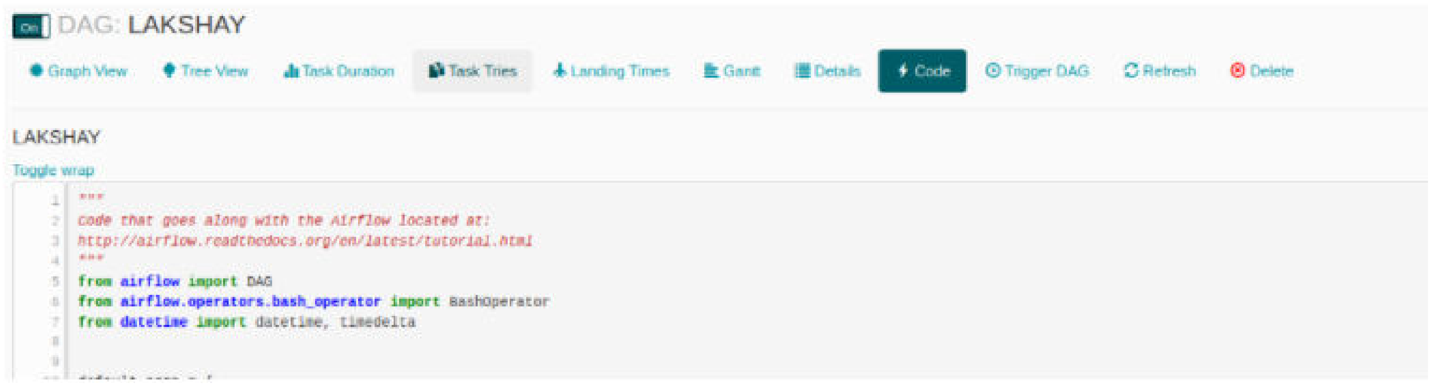


Cumulative Duration ☐

Base date: 2020-11-17 08:04:09 Number of runs: 25 Go

5) CODE:

a) In this view, you can **quickly view** the **code** that was **used** to **generate** the **DAG**.



LAKSHAY

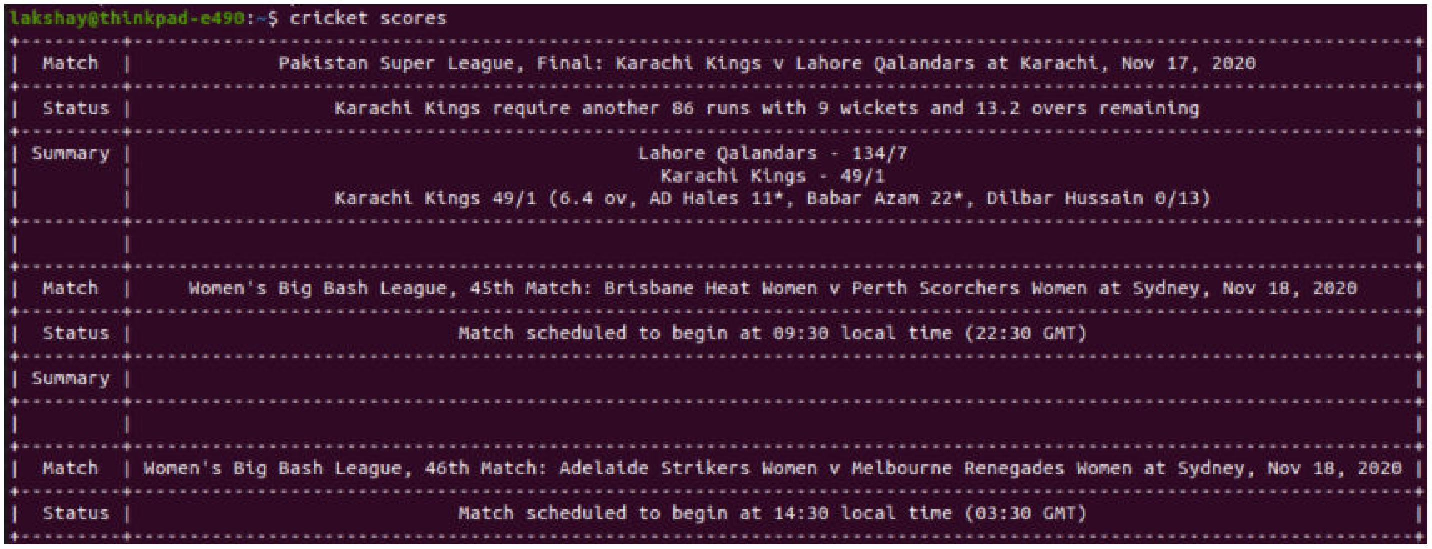
```
"""
code that goes along with the Airflow located at:
http://airflow.readthedocs.org/en/latest/tutorial.html
"""
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta


```

1) Python Operator in Apache Airflow:

An **operator** describes a **single task** of the **workflow** and **Operators** provide us **different operators** for many **different tasks** for example **BashOperator**, **PythonOperator**, **EmailOperator**, **MySqlOperator** etc.

2) BashOperator:

In this section, we will **create** a **workflow** in which the **first step** will be to **print** "Getting Live Cricket Scores" on the **terminal**, and then **using** an **API**, we will **print** the **live scores** on the **terminal**.



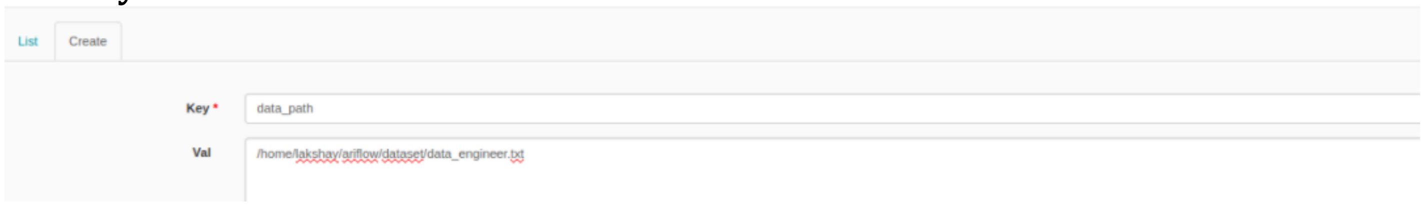
```
lakshay@thinkpad-e490:~$ cricket scores
+---------+-----------------------------------------------------------------------------------------------+
|  Match  |          Pakistan Super League, Final: Karachi Kings v Lahore Qalandars at Karachi, Nov 17, 2020          |
+---------+-----------------------------------------------------------------------------------------------+
|  Status |             Karachi Kings require another 86 runs with 9 wickets and 13.2 overs remaining             |
+---------+-----------------------------------------------------------------------------------------------+
| Summary |                                   Lahore Qalandars - 134/7                                      |
|         |                                     Karachi Kings - 49/1                                        |
|         |            Karachi Kings 49/1 (6.4 ov, AD Hales 11*, Babar Azam 22*, Dilbar Hussain 0/13)           |
+---------+-----------------------------------------------------------------------------------------------+
|         |                                                                                               |
+---------+-----------------------------------------------------------------------------------------------+
|  Match  |   Women's Big Bash League, 45th Match: Brisbane Heat Women v Perth Scorchers Women at Sydney, Nov 18, 2020   |
+---------+-----------------------------------------------------------------------------------------------+
|  Status |                    Match scheduled to begin at 09:30 local time (22:30 GMT)                    |
+---------+-----------------------------------------------------------------------------------------------+
| Summary |                                                                                               |
+---------+-----------------------------------------------------------------------------------------------+
|         |                                                                                               |
+---------+-----------------------------------------------------------------------------------------------+
|  Match  | Women's Big Bash League, 46th Match: Adelaide Strikers Women v Melbourne Renegades Women at Sydney, Nov 18, 2020 |
+---------+-----------------------------------------------------------------------------------------------+
|  Status |                    Match scheduled to begin at 14:30 local time (03:30 GMT)                    |
+---------+-----------------------------------------------------------------------------------------------+
```

What are Variables in Apache Airflow?

- We know that **Airflow** can be **used** to **create** and **manage complex workflows** and we can **run multiple workflows** at the **same time**.
- There is a **possibility** that **most** of your **workflows** are **using** the **same database** or **same file path**.
- Now, if we make **any changes** like **changing** the **path** of the **directory** where we **save files** or **change** the **configuration** of the **databases**.
- In that case, you **don't** want to **go** and **update each** of the **DAGS separately**.
- Airflow **provides** a **solution** for this, you **can create variables** where you **can store** and **retrieve data** at **runtime** in the **multiple DAGS**.
- So if any **major changes occur**, you can **just edit your variable** and **your workflows** are **good to go**.

How to create Variables?

- Open the Airflow **dashboard** and click on the **Admin** from the **top menu** and then click on **Variables**.
- Now, click on **Create** to create a new variable and a window will open.
- Add the **key** and **value** and submit it.
- Here, I am **creating** a **variable** with the **key** name as **data_path** and **value** as the **path** of **any** random **text file**.



Key *	data_path
Val	/home/lakshay/ariflow/dataset/data_engineer.txt

Now, we will create a DAG where we will find out the word count of the text data present in this file. When you want to use the variables, you need to import it.

4) MySQLOperator:

5) EmailOperator: