

# Final Task Assessment



By - Ashish Gusai  
Company ID - 10201  
Mentors - Chetan Khatri  
& Babu Prabhakar

# List of Tasks



1. Add Airflow on Ambari HDP using mpack.
2. Work with Local executor ,Sequential Executor ,Celery Executor.
3. Work with BranchPythonOperator
4. PartitionBy work with multiple category ,repartition and apply filters ,aggregation ,groupBy ,orderBy on dataframes.
5. Using master=local instead of YARN
6. Work with kaggle datasets and apply some business logic to get meaningful data.
7. Upload code and presentation at GitHub.



# Airflow using MPack

- What is MPack?
  - add an "add-on" service (custom service) to the stack definition, one has to manually add the add-on service to the stack definition on an Ambari Server. There is no release vehicle that can be used to ship add-on services.
- Problem Statement :- Airflow webserver UI is not responding.
- Solution :- No solution found yet.



# Airflow Sequential Executor

- Airflow uses a sqlite database, which you should outgrow fairly quickly since no parallelization is possible using this database backend. It works in conjunction with the SequentialExecutor which will only run task instances sequentially.



# Airflow Local executor

- LocalExecutor executes tasks locally in parallel. It uses the multiprocessing Python library and queues to parallelize the execution of tasks.
- Need to install PostgreSQL or MySQL to support parallelism.



# Airflow Celery Executor

- CeleryExecutor is one of the ways you can scale out the number of workers. For this to work, you need to setup a Celery backend (RabbitMQ, Redis, ...) and change your airflow.cfg to point the executor parameter to CeleryExecutor and provide the related Celery settings.

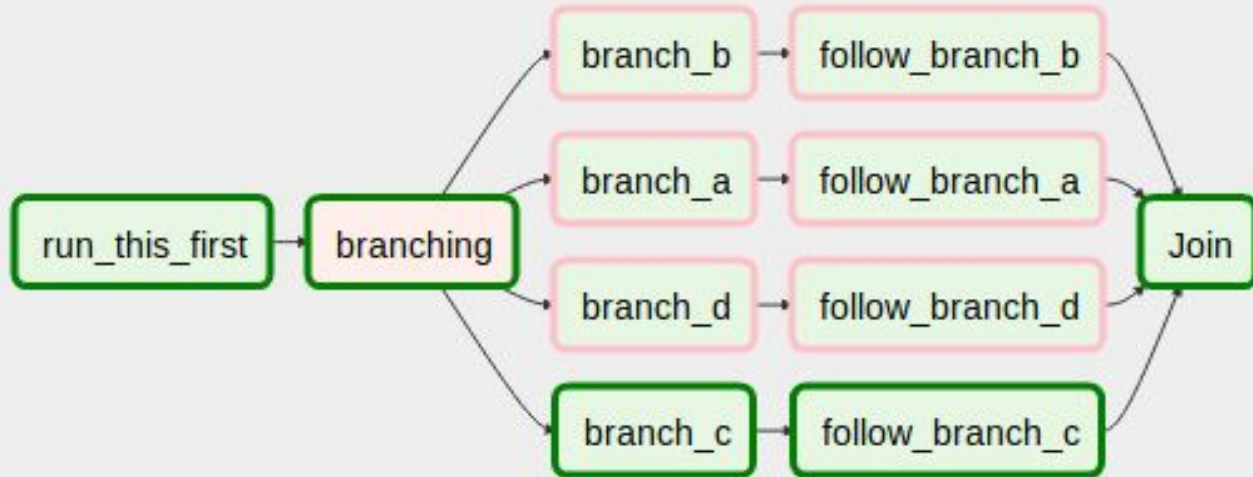


# BranchPythonOperator

- The BranchPythonOperator is much like the PythonOperator except that it expects a python\_callable that returns a task\_id. The task\_id returned is followed, and all of the other paths are skipped. The task\_id returned by the Python function has to be referencing a task directly downstream from the BranchPythonOperator task.
- Code:-[https://github.com/aashishqusai/Internship\\_assessments/blob/master/BranchPythonOperator/branch\\_python\\_op.py](https://github.com/aashishqusai/Internship_assessments/blob/master/BranchPythonOperator/branch_python_op.py)



# BranchPythonOperator







# Working with Kaggle Dataset

- Dataset:- Data Science for Good: DonorsChoose.org
- Problem Statement:-
  - DonorsChoose.org has funded over 1.1 million classroom requests through the support of 3 million donors, the majority of whom were making their first-ever donation to a public school. If DonorsChoose.org can motivate even a fraction of those donors to make another donation, that could have a huge impact on the number of classroom requests fulfilled.



# Dataset Overview

1. Donations.csv: For every project in the Projects.csv dataset, there are one or more donations. This dataset contains each donation from a citizen donor and is joined with the dataset above using the “Project ID” column.test test1
2. Donors.csv
3. Resources.csv: For every project in the Projects.csv file, there are one or more resources that is requested. This dataset contains the names of each resource in the project request and is joined with the dataset above using the “Project ID” column.
4. Projects.csv
5. Schools.csv: More information at a school level. Each row represents a single school. This dataset is joined with the project data using the “School ID” column.
6. Teachers.csv: More information at a teacher level. Each row represents a single teacher. This dataset is joined with the project data using the “Teacher ID” column.



# Programs and Dataflow

DataFlow :- [WorkFlow](#)

Code :-

[https://github.com/aashishgusai/Internship\\_assessments/tree/master/DonorsChooseCode](https://github.com/aashishgusai/Internship_assessments/tree/master/DonorsChooseCode)



# YAML Encryption

- For encrypting YAML i have used yaycl-crypt 0.4.0 library.
- It uses python Cryptography libraries for encryption.

Code:-

[https://github.com/aashishgusai/Internship\\_assessments/blob/master/DonorsChooseCode/db\\_conf.py](https://github.com/aashishgusai/Internship_assessments/blob/master/DonorsChooseCode/db_conf.py)



# References

- MPack-Ambari:-<https://medium.com/@mykolamykhalov/integrating-apache-airflow-with-apache-ambari-ccab2c90173>
- <https://github.com/miho120/ambari-airflow-mpack>
- <https://medium.com/@mykolamykhalov/integrating-apache-airflow-with-apache-ambari-ccb2c90173>
- <https://medium.com/a-r-g-o/installing-apache-airflow-on-ubuntu-aws-6ebac15db211>
- Sequential Executors:-<https://airflow.apache.org/start.html>
- Local executors:-[http://airflow.apache.org/\\_modules/airflow/executors/local\\_executor.html](http://airflow.apache.org/_modules/airflow/executors/local_executor.html)
- Celery Executor:-<https://airflow.apache.org/howto/executor/use-celery.html>
- BranchPythonOperator:-<https://airflow.apache.org/concepts.html>
- Kaggle Dataset:-<https://www.kaggle.com/donorschoose/io/home>
- Yaml encryption:- <https://pypi.org/project/yaycl-crypt/>