
Jagoan
Hosting_
<#YangPentingYakin/>
WORKFLOW USING
AIRFLOW_
29 / 02
2020

ABOUT ME

1. **Raja Fathurrahman**
2. I am **Physicist** (Computational and Instrumentation physics)
3. Data Analyst at **JagoanHosting**

What is workflow ?

Workflows are the way process get work done, and can be illustrated as series of steps that need to be completed sequentially in a diagram or checklist.

- Sequence of **Tasks**
- Started on **schedule** or **trigger** by event
- Frequently used to handle data **pipelines**

Jagoan Hosting_

Typical and Example what we can do with workflow

- ETL (Extract, Transform, Load)
- Data Warehousing
- A/B Testing
- Anomaly Detection
- Training Recommendation testing
- Orchestrating automated process

Jagoan Hosting_

Workflow Manager

argo

Azkaban

Apache Airflow

luigi

OOZIE

Jagoan Hosting_

Open-source workflow orchestration tools and comperration

Tool	Paradigm	Workflow syntax/DSL	UI	Origin	Platform(s)	Popularity
Airflow	DAG configuration as code	Python	Mature UI	AirBnB, Apache	*nix	13481 Stars
Argo Workflows	Tasks described as containers	YAML-based templates	Mature UI	Intuit	Kubernetes	3521 Stars
Azkaban	Declarative with DSL	Flow 2.0 (YAML-like)	Mature UI	LinkedIn (Hadoop)	*nix	2727 Stars
Luigi	Object oriented (Task, Target)	Python	Mature UI	Spotify	*nix, Windows	12063 Stars
Oozie	Declarative with DSL	XML-based	Mature UI	Hadoop-only, Apache	*nix, Java	500 Stars

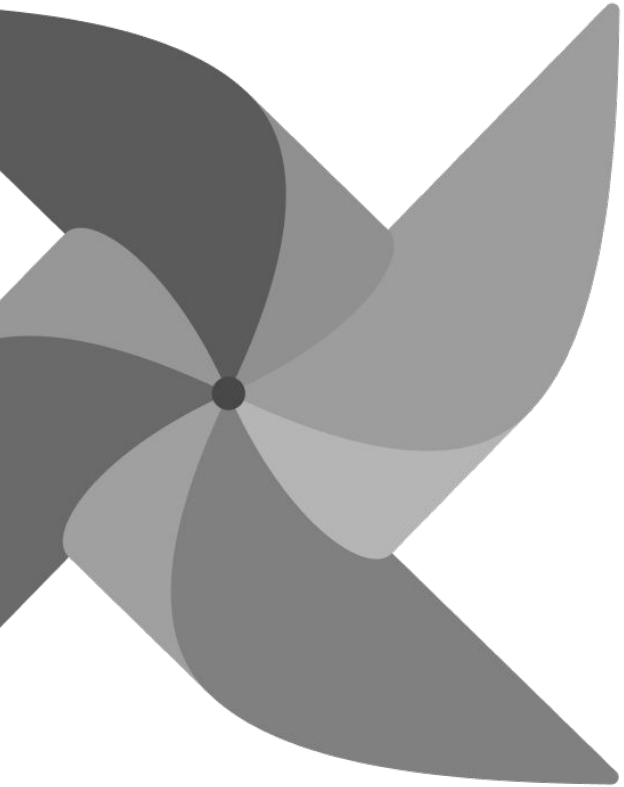
Why Apache airflow ?

We've also observed a general shift away from drag-and-drop ETL (Extract Transform and Load) tools towards a more programmatic approach. Product know-how on platforms like Informatica, IBM Datastage, Cognos, AbInitio or Microsoft SSIS isn't common amongst modern data engineers, and being replaced by more generic software engineering skills along with understanding of programmatic or configuration driven platforms like Airflow, Oozie, Azkabhan or Luigi. It's also fairly common for engineers to develop and manage their own job orchestrator/scheduler.

- ‘Maxime Beauchemin’

Airflow is a workflow scheduler to help with scheduling complex workflows and provide an easy way to maintain them

Jagoan Hosting_



Introduction apache airflow

- **Programmatically** author
- **Schedule** and **Monitor** workflows
- More **maintainable**
- **Versionable**
- **Testable**
- **Collaborative**
- **Open source**
- Rich **Web UI** for **Monitoring** and **Log**

Jagoan Hosting_

How to install apache airflow

1

```
$ python3 --version
Python 3.6.0
$ virtualenv --version
15.1.0
```

2

```
$ cd /path/to/my/airflow/workspace
$ virtualenv -p `which python3` venv
$ source venv/bin/activate
(venv) $
```

3

```
(venv) $ pip install airflow==1.8.0
```

4

```
(venv) $ airflow version
  ____________       _____________
 ____    |__( )_________  __/__  /________      __
____  /| |_  /__  ___/_  /_ __  /_  __ \_ | /| / /
___  ___ |  / _  /    _  __/ _  / / /_/ /_ |/ |/ /
 _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
   v1.8.0rc5+apache.incubating
```

5

```
(venv) $ cd /path/to/my/airflow/workspace
(venv) $ mkdir airflow_home
(venv) $ export AIRFLOW_HOME=`pwd`/airflow_home
```

6

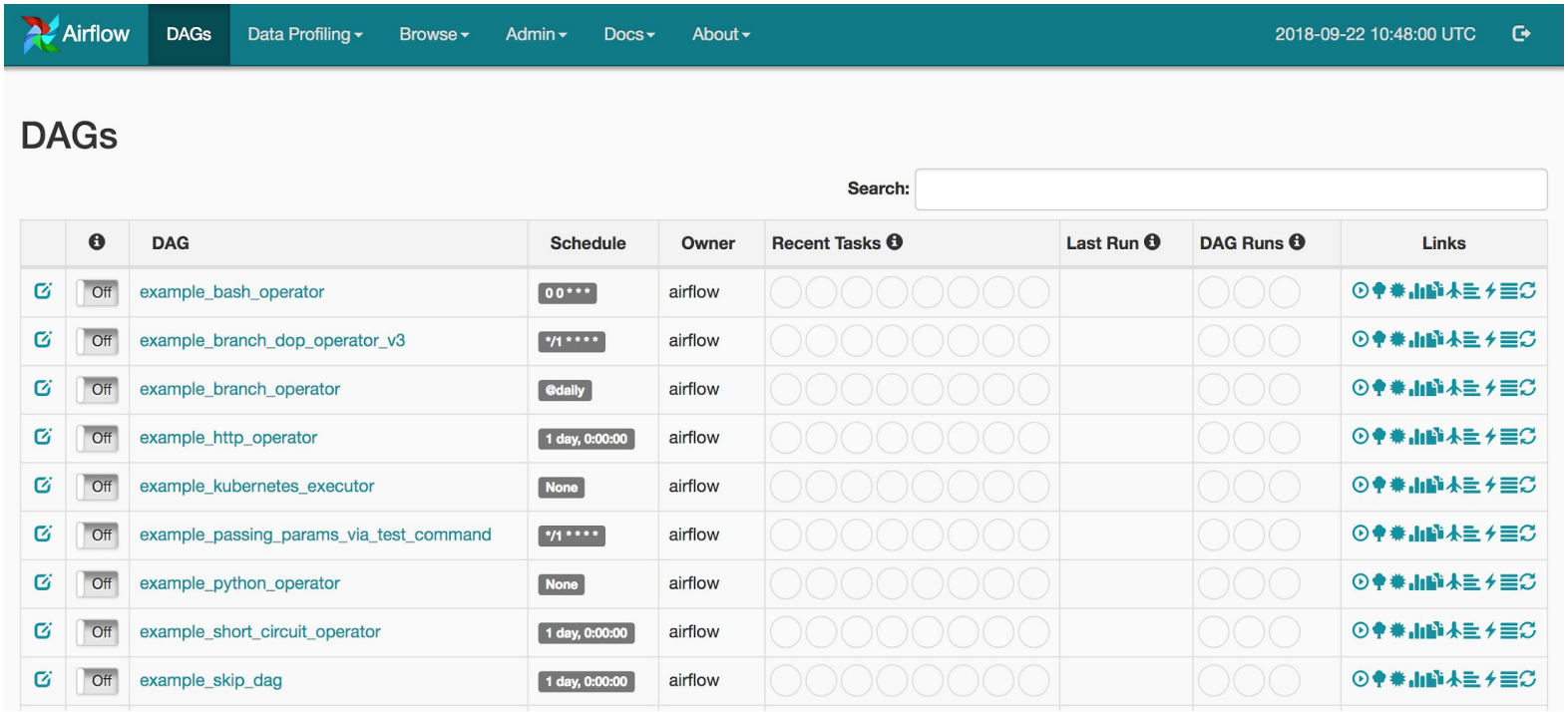
```
(venv) $ airflow initdb
```

7

```
(venv) $ airflow webserver
```

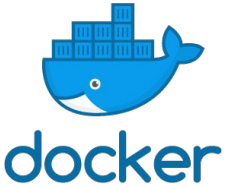
Jagoan Hosting_

Airflow UI



Jagoan Hosting_

Alternative installing apache airflow



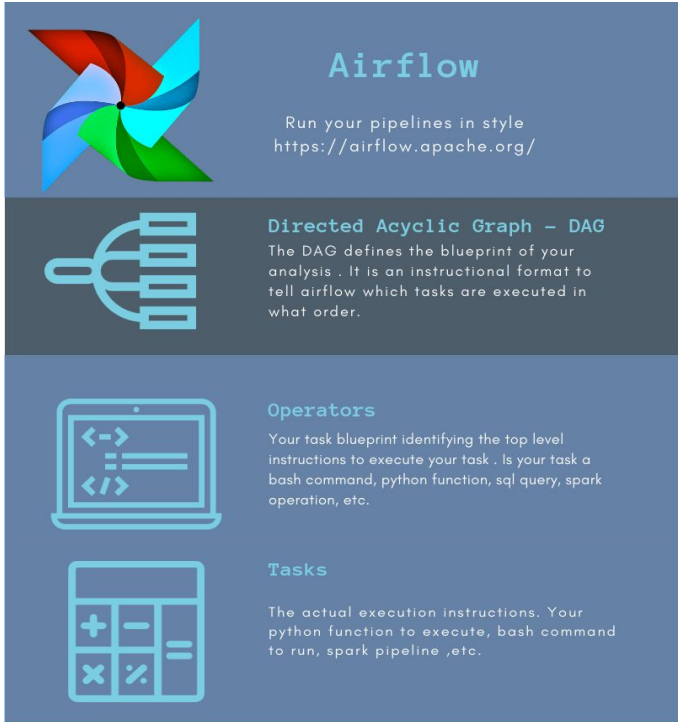
More easy installing airflow using docker just pull the images and run the container on docker and no need more configuration

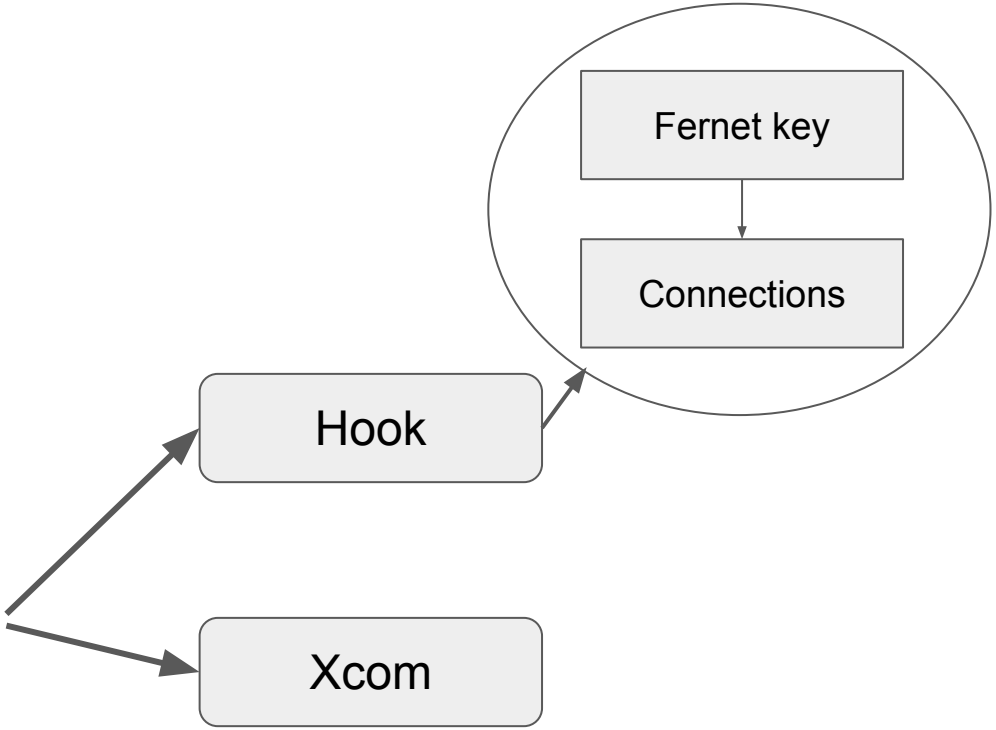
```
docker pull puckel/docker-airflow
```

```
docker run -d -p 8080:8080 puckel/docker-airflow webserver
```

Jagoan Hosting_

Workflow component in Airflow

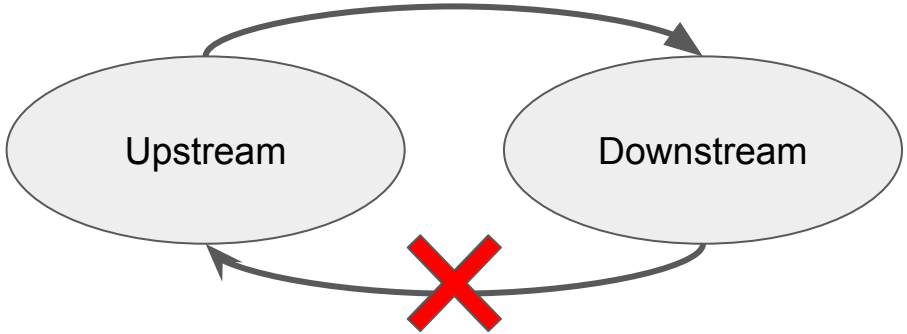




Jagoan Hosting_

What workflow in Airflow?

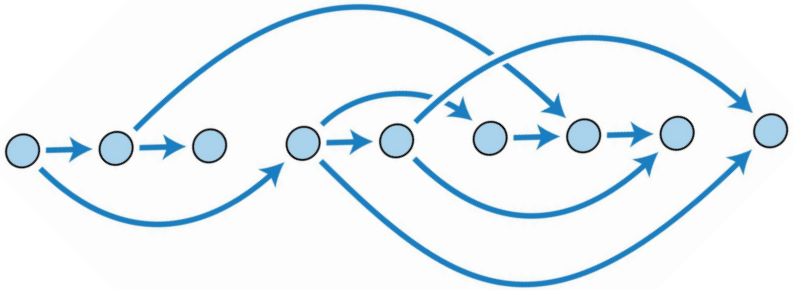
This flow like a river flow from upstream to downstream



Jagoan Hosting_

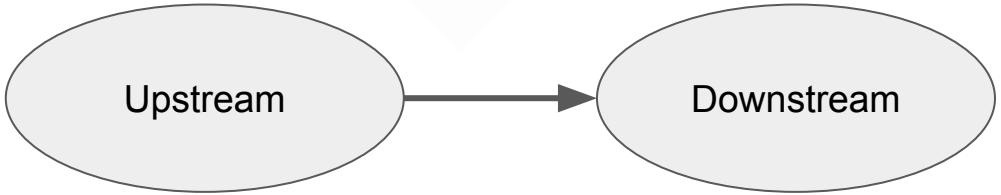
Workflow with Airflow

DAG concept using river flow analogy that is directed and not in cycle



DAG is the workflow in airflow

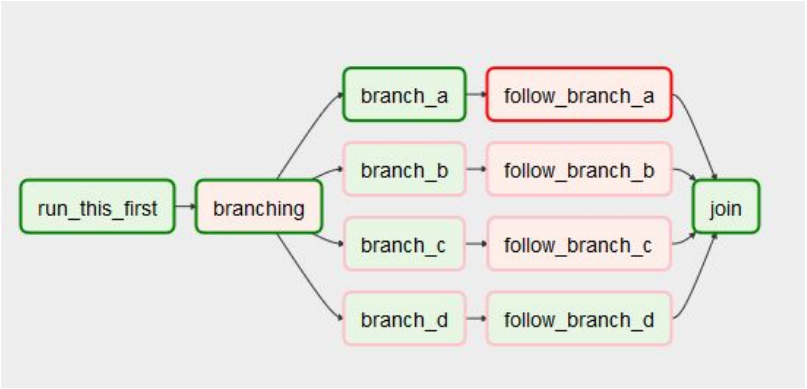
This is call a DAG (Directed Acyclic Graph)



Jagoan Hosting_

Workflow with Airflow

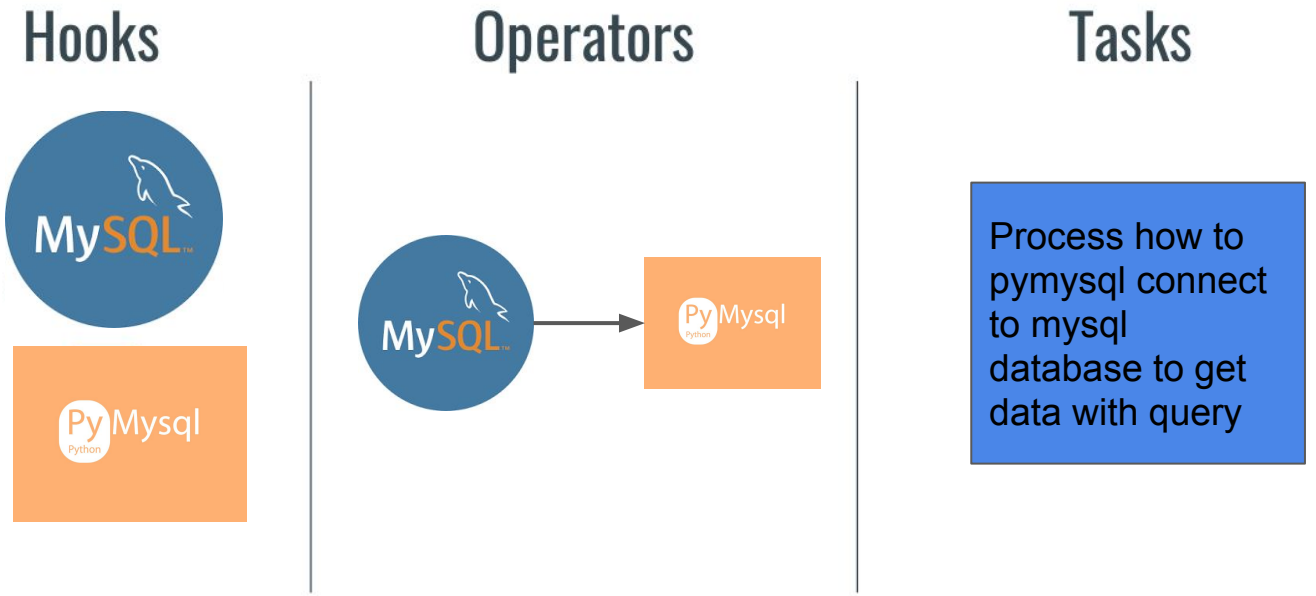
Like a river, they have a tributaries



In Airflow it call branch

Jagoan Hosting_

Workflow with apache Airflow



Jagoan Hosting_

Airflow Operators

- PythonOperator
- BashOperator
- EmailOperator
- S3ToRedshiftTransfer
- RedshiftToS3Transfer
- BigQueryToCloudStorageOperator
- GoogleCloudStorageToBigQueryOperator
- SlackAPIPostOperator
- MySqlToHiveTransfer
- SimpleHttpOperator
- DataprocOperationBaseOperator
- JiraOperator
- DockerOperator
- PostgresOperator

and many more...

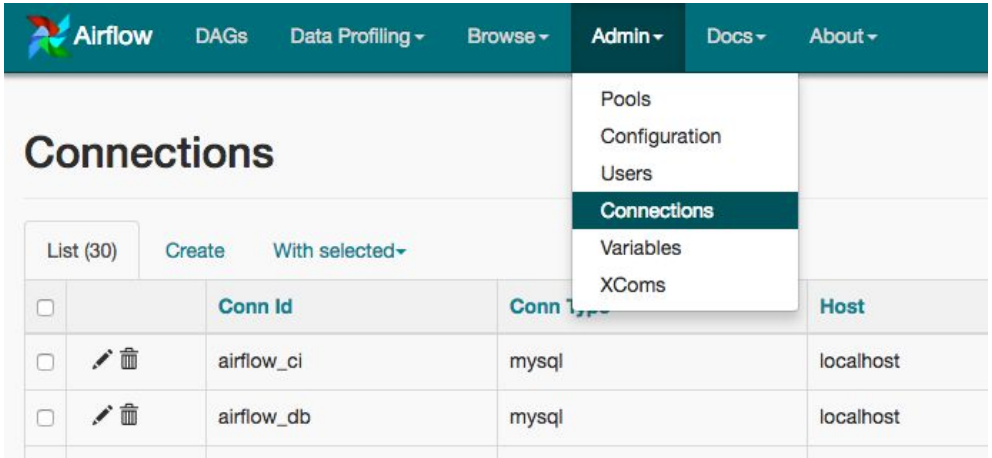
Airflow Hooks

- AwsHook
- JdbcHook
- GoogleCloudStorageHook
- CassandraHook
- BigQueryHook
- RedisHook
- DatadogHook
- AzureFileShareHook
- GrpcHook
- JiraHook
- SSHHook
- HiveCliHook
- SlackHook
- DockerHook
- S3Hook
- MySqlHook
- PrestoHook

and many more...

Connections (Hook) on Airflow

reduce connection initialization in code



Connection Types

- Amazon Web Services Connection
- Google Cloud Platform Connection
- Google Cloud SQL Connection
- gRPC
- MySQL Connection
- Oracle Connection
- PostgresSQL Connection
- SSH Connection

Jagoan Hosting_

Fernet key is important on Airflow connection

Cryptography concept on fernet key to secure your connection

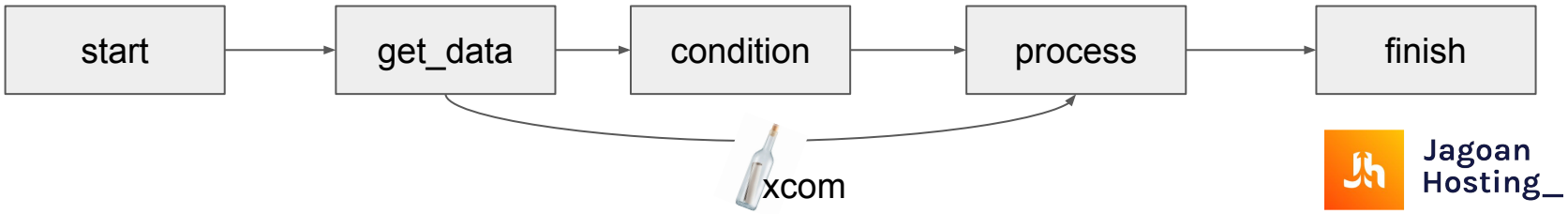
Static or Dynamic fernet key depend on yours

Jagoan Hosting_

Workflow with apache Airflow

Xcom communication between operator

Xcom concept like a message in the bottle, it can be pull and push between tasks instance, and save the data as a pickle object and best suited for small pieces of data



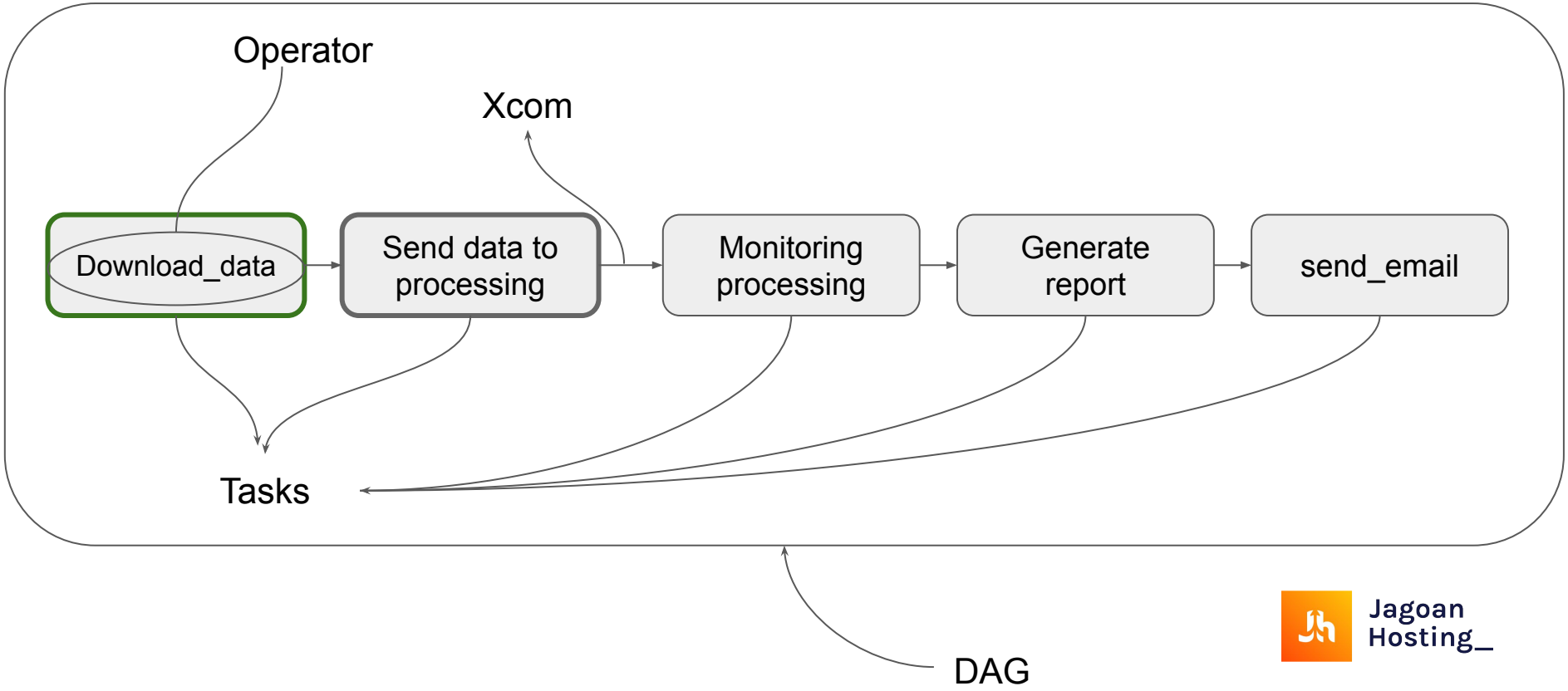
Jagoan Hosting_

Workflow with apache Airflow

preset	meaning	cron
None	Don't schedule, use for exclusively "externally triggered" DAGs	
@once	Schedule once and only once	
@hourly	Run once an hour at the beginning of the hour	0 * * * *
@daily	Run once a day at midnight	0 0 * * *
@weekly	Run once a week at midnight on Sunday morning	0 0 * * 0
@monthly	Run once a month at midnight of the first day of the month	0 0 1 * *
@yearly	Run once a year at midnight of January 1	0 0 1 1 *

Single DAG Run will be created, with an execution_date of 2016-01-01, and the next one will be created just after midnight on the morning of 2016-01-03 with an execution date of 2016-01-02.

Workflow with apache Airflow

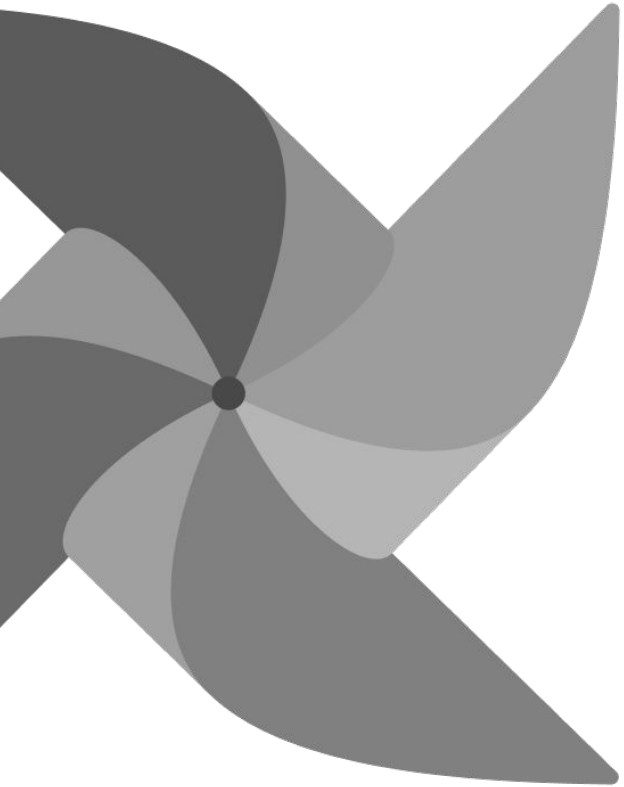


Jagoan Hosting_

What value does Airflow add?

- Retries tasks elegantly
- Alerts on failure
- Re-run specific tasks in the DAG
- Supports distributed execution

https://airflow.apache.org/docs/stable/concepts.html



Demo

Jagoan Hosting_

printHelloworld.py /home/adib/jagoanhosting/airflow/dags/printHelloworld.py/ finishProcess

```python
#Basic Airflow Library
from airflow import DAG
from airflow.operators.python_operator import PythonOperator

from datetime import datetime, timedelta, date, timezone #library for time

default_args = {
    'owner': 'Hello',
    'depends_on_past': False,
    'start_date': datetime(2020, 2, 28),
    'retries': 1,
    'retry_delay': timedelta(minutes=1) }

dag = DAG('HelloWorld_01', default_args=default_args, schedule_interval=None)

def startProcess():
    print('FINISH')

def finishProcess():
    print('FINISH')

def printHelloworld(**kwargs):
    date = kwargs['ds']
    dates='{}'.format(date)
    execute = datetime.strptime(dates, '%Y-%m-%d')
    date_now = (execute).strftime("%Y-%m-%d")

    print("Hello world!!, This is", date_now)

t1 = PythonOperator(
    task_id= 'startProcess',
    python_callable=startProcess,
    dag=dag)

t2 = PythonOperator(
    task_id= 'printHelloworld',
    provide_context=True,
    python_callable=printHelloworld,
    dag=dag)

t3 = PythonOperator(
    task_id= 'finishProcess',
    python_callable=finishProcess,
    dag=dag)

t1 >> t2 >> t3
```

Import module

DAG

Operator

Tasks

Dependencies

Jagoan Hosting_

xcom_example.py /home/adib/jagoanhosting/airflow/dags/xcom_example.py/...

```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago

args = {
    'owner': 'airflow',
    'start_date': days_ago(2)}

dag = DAG('example_xcom',default_args=args, schedule_interval="@once")

value_2 = {'a': 'b'}

def push(**kwargs):
    ti = kwargs['task_instance']
    value = [1, 2, 3]
    ti.xcom_push(key='value_1', value=value)

def push_by_returning(**kwargs):
    return value_2

def puller(**kwargs):
    ti = kwargs['ti']

    pulled_value_1 = ti.xcom_pull(key='value_1', task_ids='push')
    print('Xcom get value_1 : ',pulled_value_1)
    pulled_value_2 = ti.xcom_pull(task_ids='push_by_returning')
    print('Xcom get values_2',pulled_value_2)

push1 = PythonOperator(
    task_id='push',
    python_callable=push,
    provide_context=True,
    dag=dag)

push2 = PythonOperator(
    task_id='push_by_returning',
    python_callable=push_by_returning,
    dag=dag)

pull = PythonOperator(
    task_id='puller',
    python_callable=puller,
    provide_context=True,
    dag=dag)

[push1, push2] >> pull
```

Xcom push

Xcom pull

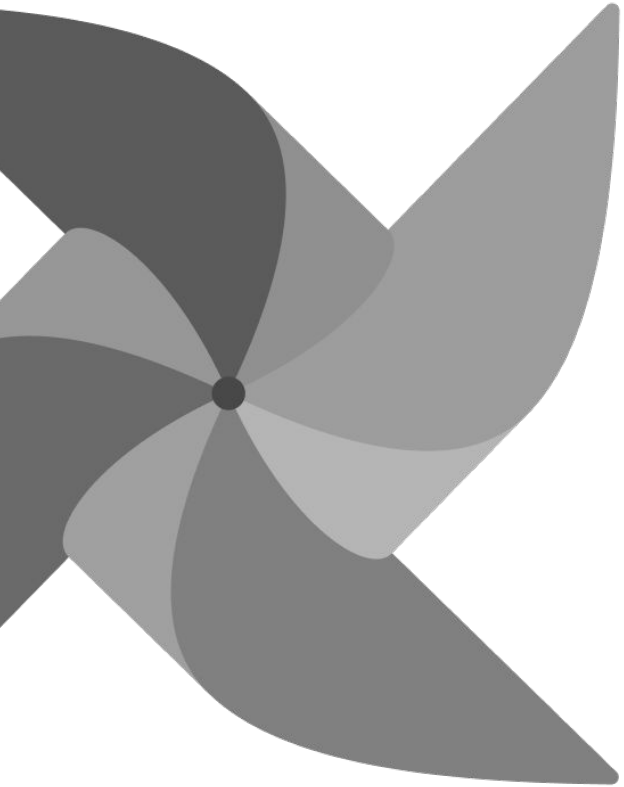
Jagoan Hosting_

Xcom Demo

```python
def push(**kwargs):
    """Pushes an XCom without a specific target"""
    ti = kwargs['task_instance']
    value = [1, 2, 3]
    ti.xcom_push(key='value_1', value=value)
```

```python
def puller(**kwargs):
    """Pull all previously pushed XComs and check if the pushed values match the pulled values."""
    ti = kwargs['ti']
    # get value_1 with key value pair
    pulled_value_1 = ti.xcom_pull(key='value_1', task_ids='push')
    print('Xcom get value_1 : ',pulled_value_1)
    # get value_2 without key value
    pulled_value_2 = ti.xcom_pull(task_ids='push_by_returning')
    print('Xcom get values_2',pulled_value_2)
```

Jagoan Hosting_



Web UI (User Interface)

Jagoan Hosting_

JADI TUA ITU PASTI,
JADI KEREN ITU PILIHAN

Mulai langkah Keren-mu dari Sekarang di :

bit.ly/skuymagang

www.jagoanhosting.com

Jagoan Hosting

Raja Fathurrahman
Data Analyst

Contact :
083129031420
raja@jagoanhosting.com