Introduction to Apache Airflow

Programmatically Manage Your Workflows for Data Engineering

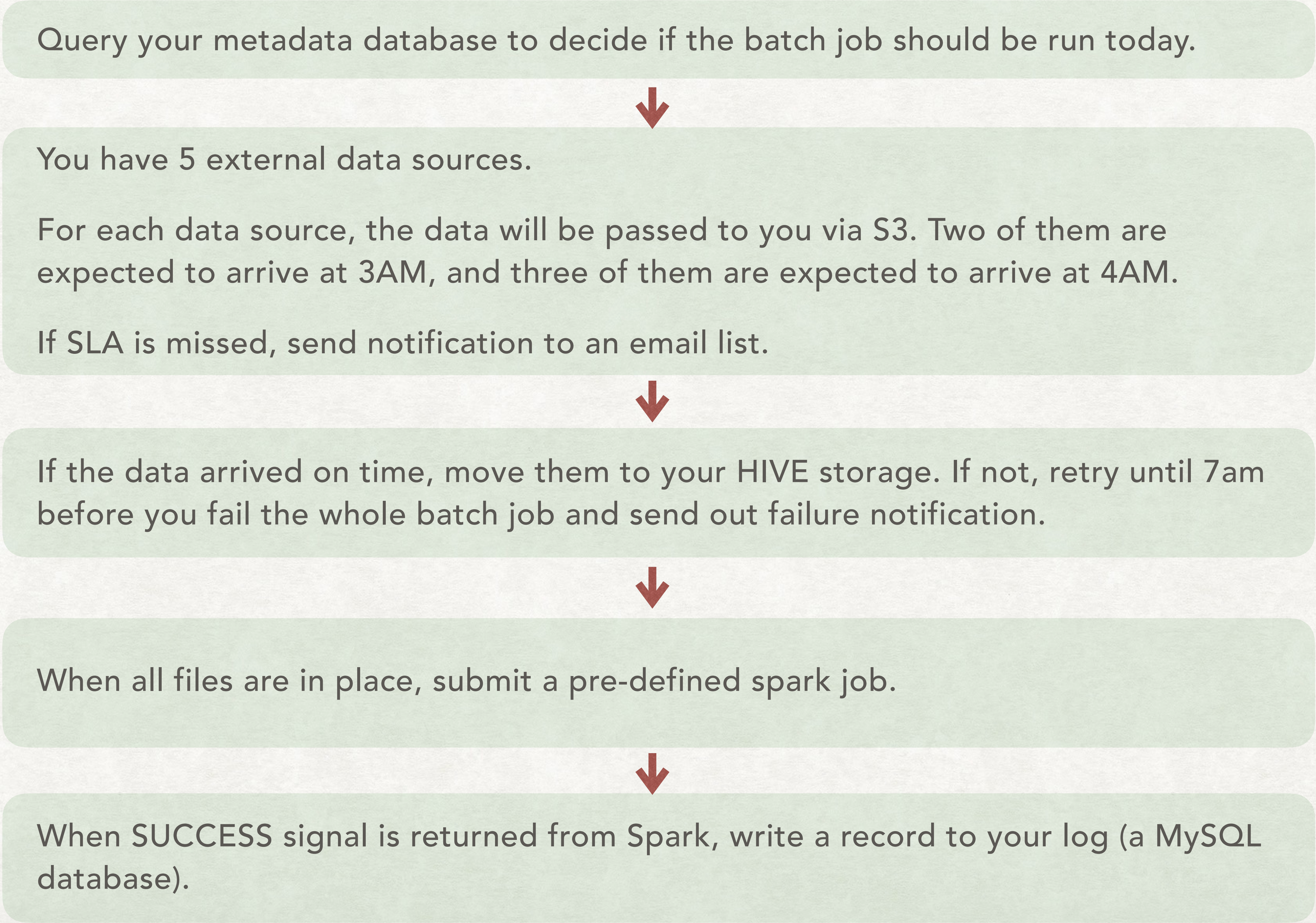
Xiaodong DENG
XD-DENG.com

August 2018

SHORT-BIO

- ***Education***
  - M.Sc in Mathematics, National University of Singapore, Singapore (2014-2016)
  - B.Sc in Applied Mathematics, Beijing Forestry University, China (2010-2014)
- ***Working Experience***
  - **2018 July - Present:** Data Engineer, DBS Bank
  - **2017 May - 2018 June:** Assistant Manager, Advanced Analytics, Manulife Insurance
  - **2016 April - 2017 April:** Data Analytics Specialist, AXA Insurance

LET'S IMAGINE - A VERY SIMPLE USE CASE



Query your metadata database to decide if the batch job should be run today.

You have 5 external data sources.

For each data source, the data will be passed to you via S3. Two of them are expected to arrive at 3AM, and three of them are expected to arrive at 4AM.

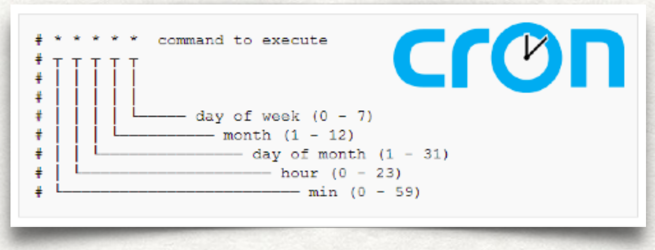
If SLA is missed, send notification to an email list.

If the data arrived on time, move them to your HIVE storage. If not, retry until 7am before you fail the whole batch job and send out failure notification.

When all files are in place, submit a pre-defined spark job.

When SUCCESS signal is returned from Spark, write a record to your log (a MySQL database).

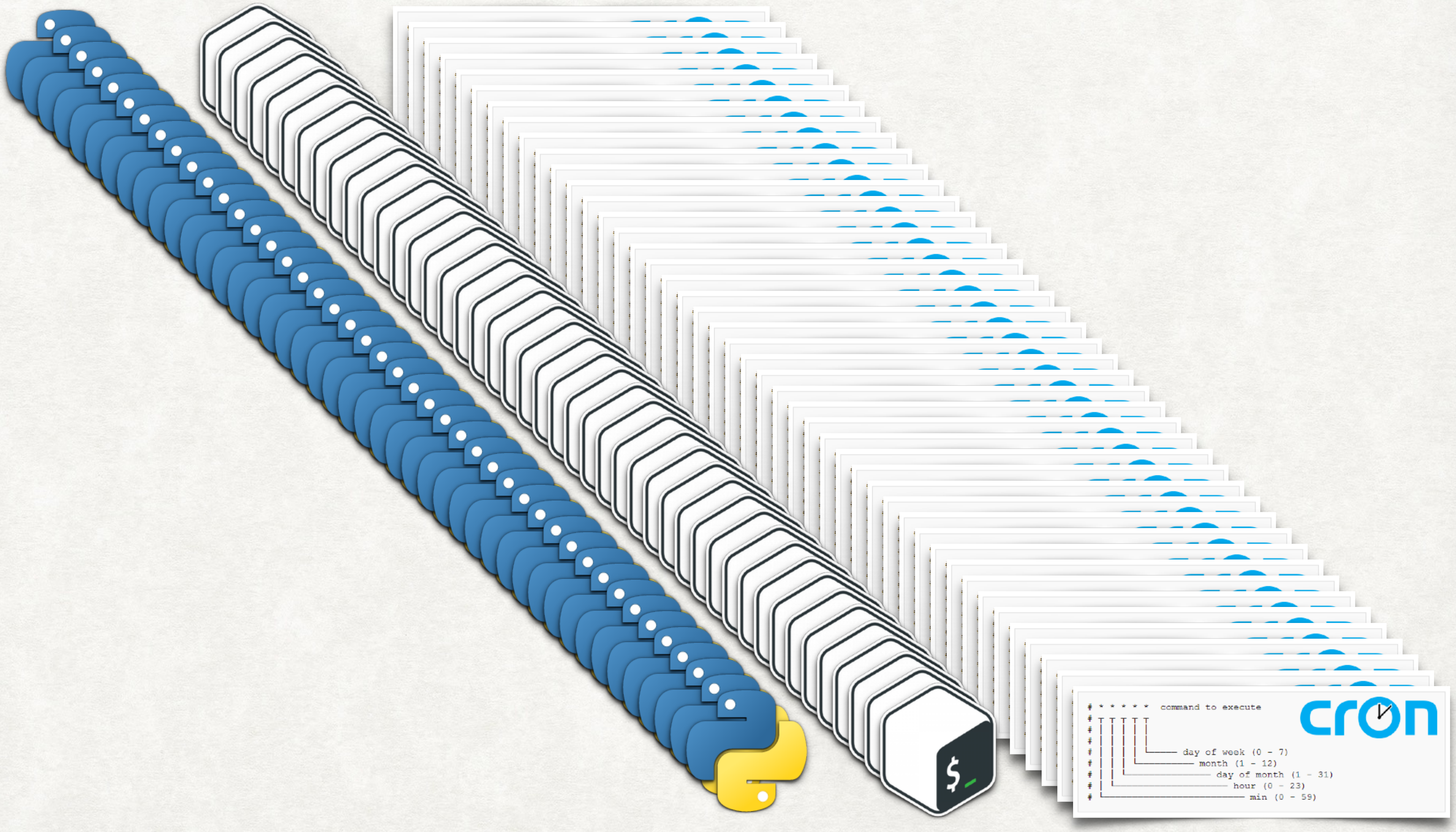
LET'S IMAGINE - A VERY SIMPLE USE CASE



```
# * * * * *  command to execute
# ┬ ┬ ┬ ┬ ┬
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ └───── day of week (0 - 7)
# │ │ │ └────────── month (1 - 12)
# │ │ └─────────────── day of month (1 - 31)
# │ └──────────────────── hour (0 - 23)
# └───────────────────────── min (0 - 59)
```

"Scripting + Cron would do!"

LET’S IMAGINE - A VERY “SIMPLE” USE CASE



```
# * * * * *  command to execute
# ┬ ┬ ┬ ┬ ┬
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ └───── day of week (0 - 7)
# │ │ │ └────────── month (1 - 12)
# │ │ └─────────────── day of month (1 - 31)
# │ └──────────────────── hour (0 - 23)
# └───────────────────────── min (0 - 59)
```

What if you have hundreds of workflows to manage?

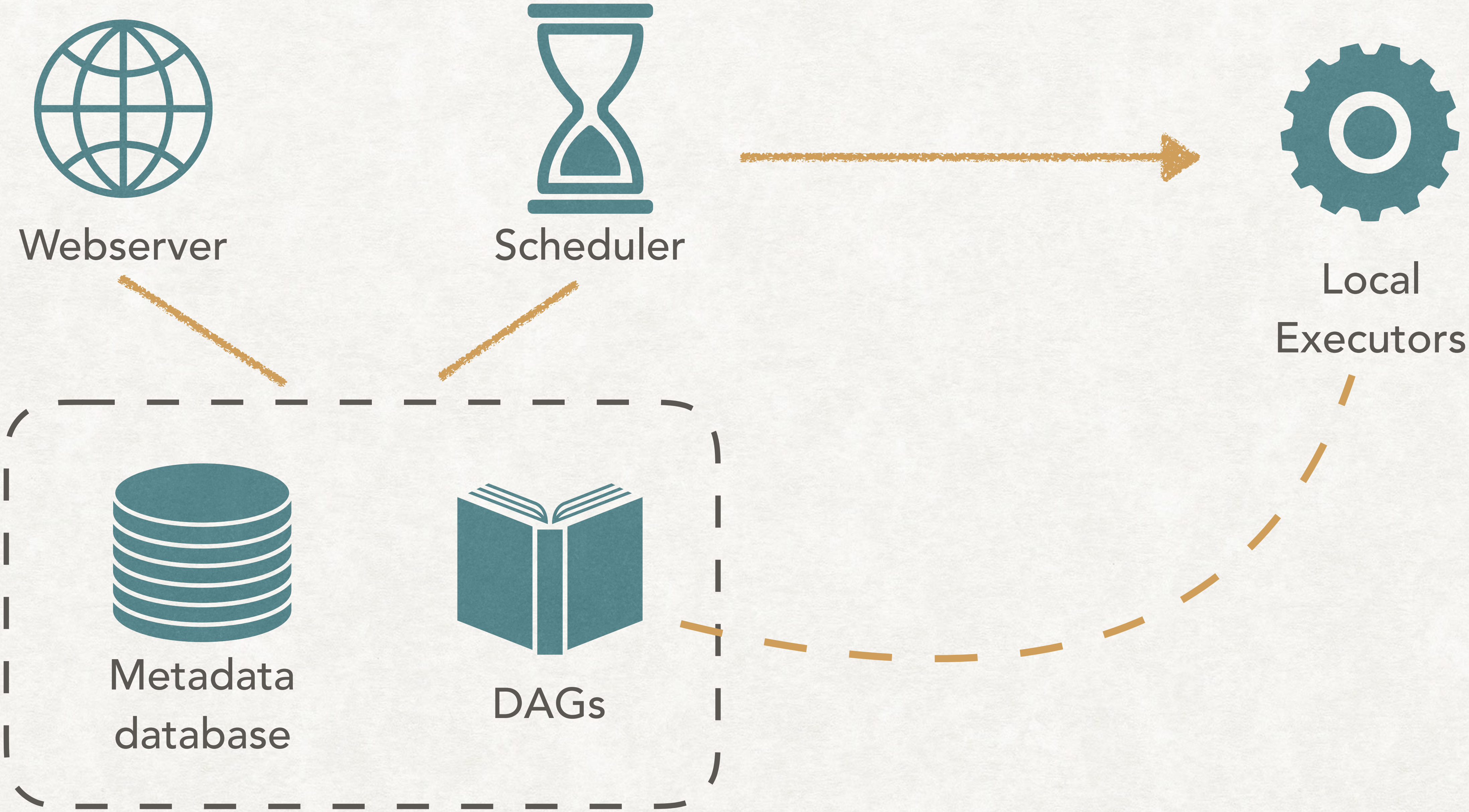
LET'S IMAGINE - A NO MORE SIMPLE USE CASE

- **Scalability in terms of managing**
  - How do you manually manage scripts & Cron expressions for hundreds of workflows?
- **Scalability in terms of execution**
  - For the consideration of performance, you may want to run your jobs on multiple worker nodes, how do you manage them?
- **Environment Dependencies**
  - Different jobs may have different dependencies, e.g., Spark, or network proxy, etc.
- **Connections to different systems (like *RDBMS*, *AWS*, *Hive*, *HDFS*, etc)**
  - like *RDBMS*, *AWS*, *Hive*, *HDFS*, etc. All of them come together with configurations like host address, port, id, password, schema, etc. How to manage them in a centralised fashion?
- **Monitoring**
  - How do we monitor the status of each step? Which batch job failed? Due to which step? For what reason?
- **Re-running**
  - How can we re-run a specific step? Manually do it or make ad-hoc change on the script? Neither is ideal.

APACHE AIRFLOW (INCUBATING)

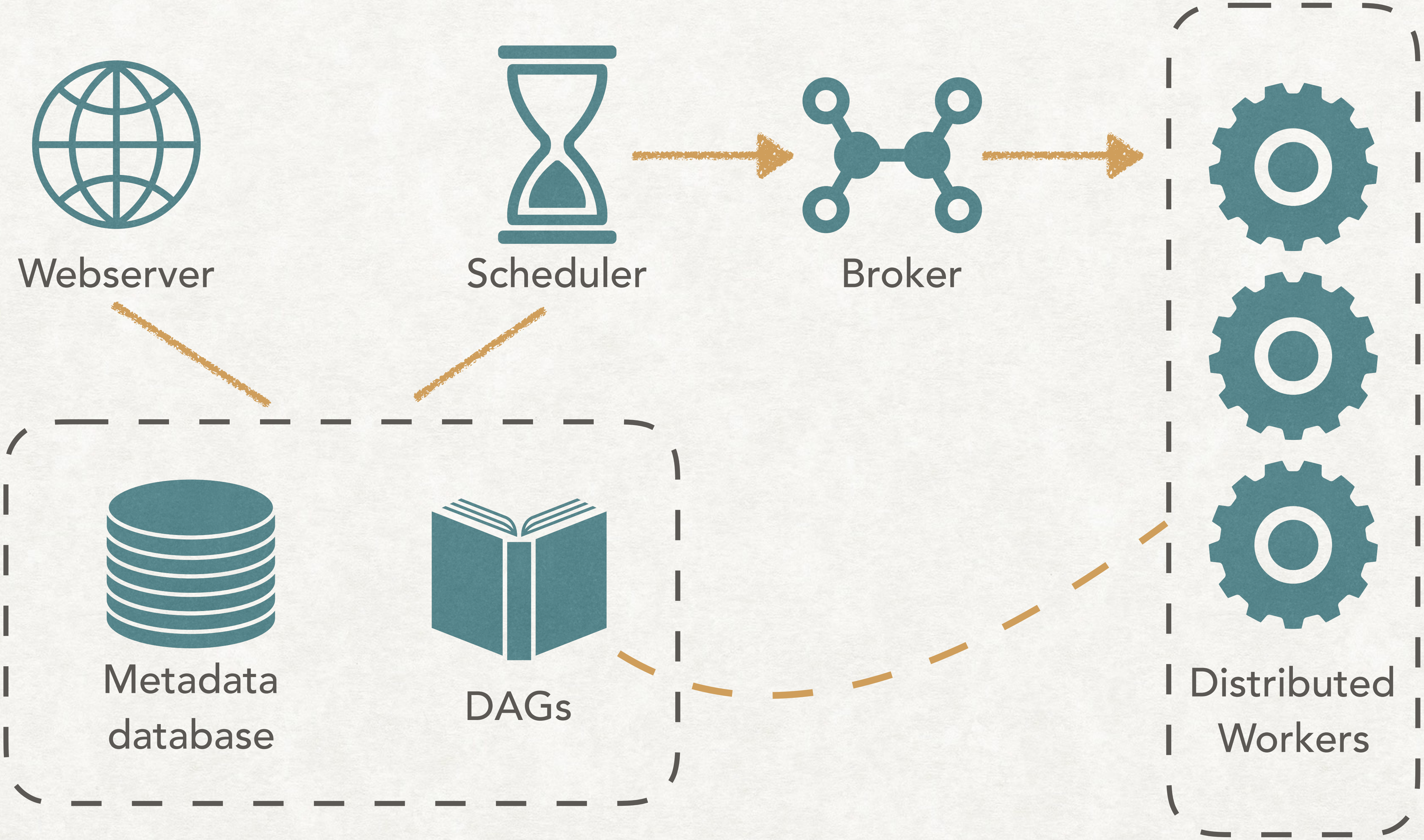
- Started in 2014 at Airbnb
- Became an Apache incubator project in 2016
- Written in Python
- 500+ contributors (according to GitHub history)
- A platform to programmatically author, schedule and monitor workflows
- Workflows are defined as directed acyclic graphs (DAG) and configured as Python scripts.
- Supports distributed execution
- Friendly interface

APACHE AIRFLOW (INCUBATING)



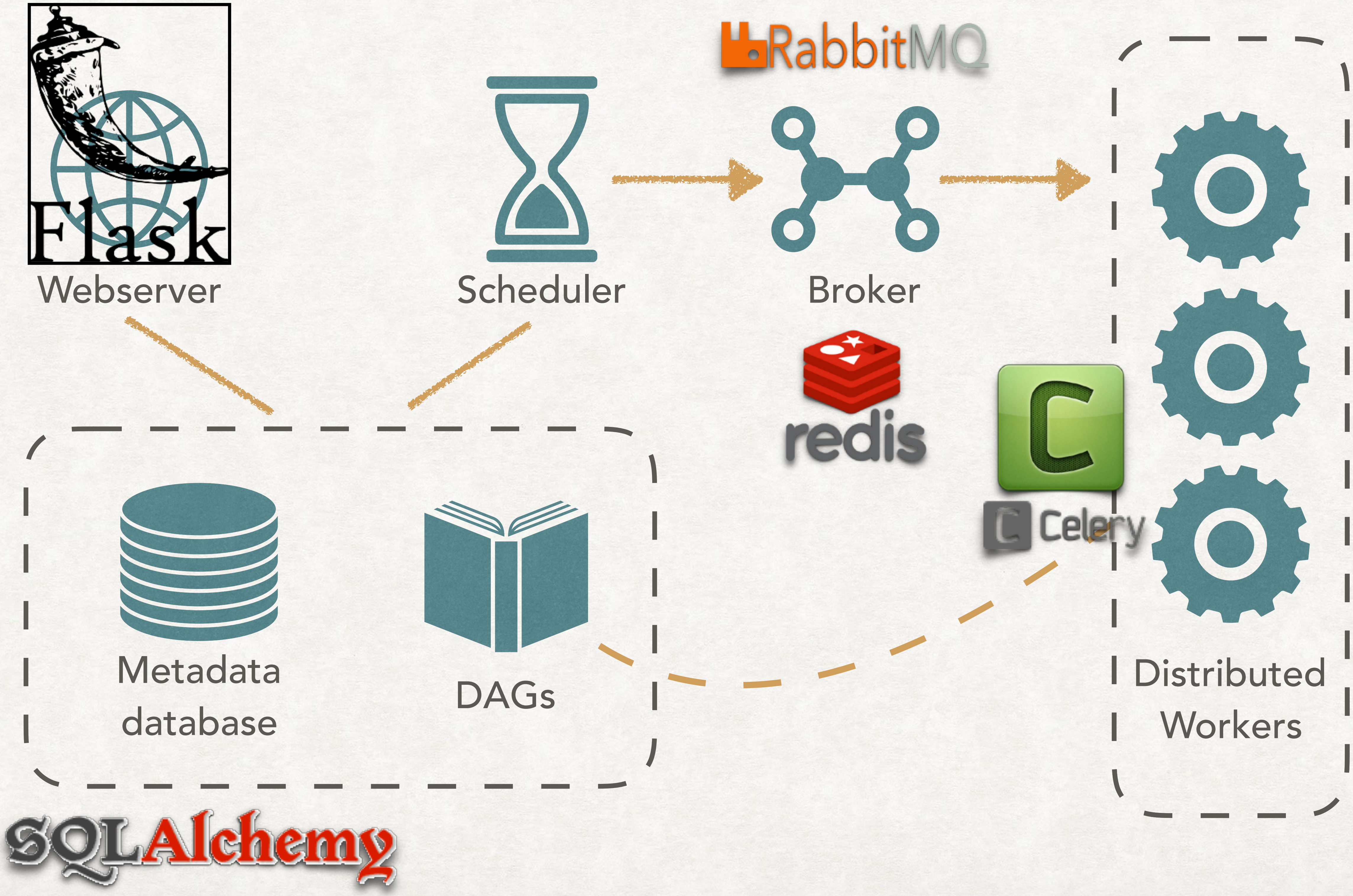
Reference: https://www.slideshare.net/sumitmaheshwari007/apache-airflow

APACHE AIRFLOW (INCUBATING)

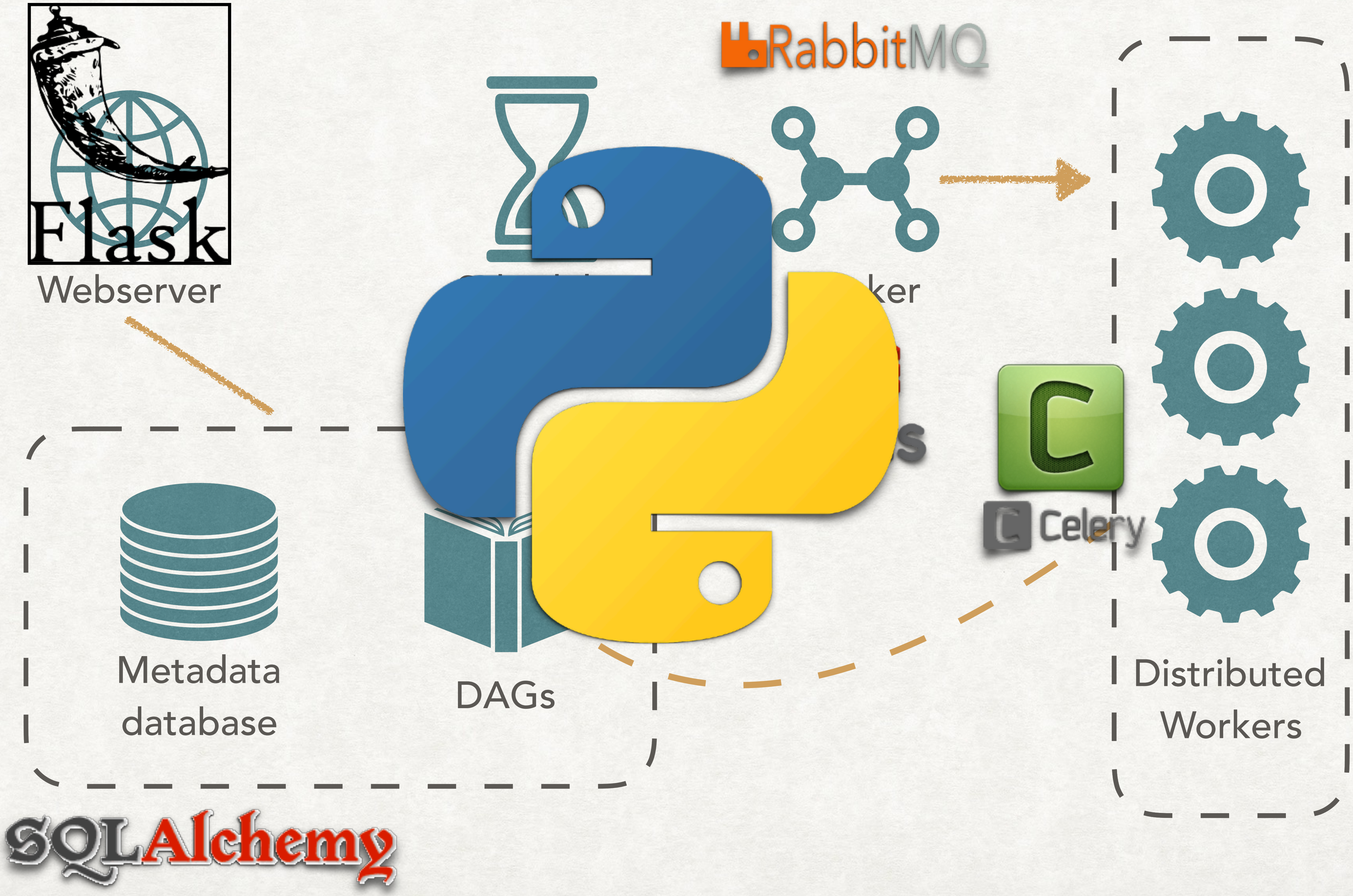


Reference: https://www.slideshare.net/sumitmaheshwari007/apache-airflow

APACHE AIRFLOW (INCUBATING)



APACHE AIRFLOW (INCUBATING)



DEMO

WHERE IS YOUR DEMO?!

Thanks!