Data lineage via Apache Airflow
Short bio Matteo

Matteo Senardi

* 2020-ongoing Head of Data in Docsity
* 2019-2020 Senior Data Engineer in Mediaset
* 2016-2019 Innovation Engineer in Webranking

* email: matteo.s@docsity.com

SCAN ME
https://github.com/pualien
Short bio Irene

Irene Soligno

* 2020-ongoing Data Scientist in Docsity

* 2019-2020 Data Scientist in Enginium

* 2015-2019 PhD and research fellow in POLITO (environmental engineering)

* email: irene.s@docsity.com

* ORCID: https://orcid.org/0000-0001-9884-5316
Founded in Italy in 2011 - Now present in 70 countries, with offices in Turin, Rome and Sao Paulo (Brazil).

Docsity is an EdTech company operating on an international scale that distributes and produces digital education content to help university students prepare for their exams.
Docsity also collaborates with over 150 universities and business schools from around the world, by promoting their educational programs through the Docsity student community (Business To Business - B2B).

Docsity has today **500K** new registered users each month and **12M** visitors per month.

Full-service approach: implement and optimize the whole student funnel.
WHY AIRFLOW?

part I
Let’s imagine - a very simple use case

01  PROBLEM

Decide if batch job should run today and how long to backfill

02  PROCESS

Data sources, that need to be passed through data lake

03  TARGET

When all files are in place, submit the pre-defined job.
Let’s imagine - a very simple use case

Scripting + Cron could do!
Let’s imagine - a very simple use case

What if hundreds of workflows to be managed?
Let’s imagine - a very simple production use case

Which features are required to manage production workflows?

**MANAGE**
Manage scripts and crontab for hundred of workflows

**ENVIRONMENT**
Jobs could have different requirements and environments

**EXECUTION**
Consider performance in parallel

**CONNECTION**
Operational DBs, APIs, Cloud Services, ssh tunneling, all with their own configuration

**MONITOR**
Track performance and completion of each step

**RETRY**
Re-run a specific step in case of failures or whatever
Apache Airflow
born in 2014 from airbnb

The technology to automate & orchestrate data pipelines

~1700 GITHUB CONTRIBUTORS
PYTHON BASED
MONITORING
DAGS AS CODE
SECURITY BY DESIGN
FRIENDLY INTERFACE
DISTRIBUTED EXECUTION
DATA LINEAGE
AIRFLOW DEPLOYMENT
Airflow Docsity solution selection

**MANAGED**

- Google Cloud Composer
- Amazon Managed Workflows for Apache Airflow
- Astronomer

**CUSTOM CONTAINER**

- Google Compute Engine
- AWS ECS
- Docker
Airflow Docsity solution selection

**Managed**
- KEYS IN HAND
- MAINTENANCE BY PROVIDER
- MANAGED SCALABILITY
- PREDEFINED AIRFLOW VERSIONS

**Custom Container**
- REPRODUCIBILITY
- TESTING
- ISOLATION
- PORTABILITY
- PRICING
- CUSTOM AIRFLOW VERSIONS
Airflow Docsity solution

**MANAGED**

- KEYS IN HAND
- MAINTENANCE BY PROVIDER
- MANAGED SCALABILITY
- PREDEFINED AIRFLOW VERSIONS

**CUSTOM CONTAINER**

- REPRODUCIBILITY
- TESTING
- ISOLATION
- PORTABILITY
- PRICING
- CUSTOM AIRFLOW VERSIONS
FROM python:3.9.9-slim-buster

ARG AIRFLOW_VERSION=2.2.3

...

ENV BUILD_COMMIT_ID=$BUILD_COMMIT_ID
Airflow basic architecture

- Webserver
- Metadata DB
- DAGs
- Scheduler
- Local / sequential executor
Airflow distributed process

Webserver

Scheduler

Broker

Celery distributed task queue

Metadata DB

DAGs

Workers
Airflow distributed engine
Production architecture
CI/CD pipeline

Google Cloud Build

- Decoupled Stages
- Repeatable
- Faster Releases
- More Robust Releases
- Failing Fast
- Better Visibility on Change
- Centralised Artifacts

via cloudbuild.yaml
CI/CD pipeline

Github → Cloud Build → Container Registry → Compute Engine
CI/CD flow

- **Github**
  - *push on production branch to trigger Google Cloud Build*

- **Cloud Build**
  - *build and push Docker image*
  - *restarts Compute Engine*

- **Container Registry**
  - *new Airflow image tagged with $COMMIT_SHA*
  - *production container updated during startup-script*

- **Compute Engine**
  - *production container update notifications*

- **Cloud Function**
  - *subscribed to build update notifications*

- **Slack**
  - *build status log*
CI/CD pipeline

cloudbuild.yaml

steps:
- name: 'gcr.io/cloud-builders/docker'
- name: gcr.io/cloud-builders/gcloud
  args: [ compute, instances, stop, --airflow-engine, --zone=eu-west-1b ]
- name: gcr.io/cloud-builders/gcloud
  args: [ compute, instances, start, --airflow-engine, --zone=eu-west-1b ]

substitutions:
  _AIRFLOW_VERSION: 2.0.2

images:
- 'gcr.io/$PROJECT_ID/airflow:latest'
- 'gcr.io/$PROJECT_ID/airflow:$COMMIT_SHA'
WHY AIRFLOW?
part II
Lot of errors and time spent
Pandas + SQLAlchemy

MariaDB

SQLAlchemy

Google Sheets via df2gspread and gspread-pandas
Pandas + SQLAlchemy is not DRY

```python
import pandas as pd
from sqlalchemy import create_engine
from sshtunnel import SSHTunnelForwarder

server = SSHTunnelForwarder(
    'myuser',
    ssh_username='myuser',
    ssh_pkey='/home/myplace/.ssh/id_rsa',
    remote_bind_address=('0.0.0.0',
                         '37017')
)
```

~15 lines of code per query/table
pip install sqlalchemy-connector

```
from alchemy_connector import SQLAlchemySession

session = SQLAlchemySession(
    host='db.example.com',
    port='21',
    user='myuser',
    key='/home/myplace/.ssh/id_rsa',
    to_port='37017',
    to_host='0.0.0.0'
)
```
2020 Q0

Boost for less time spent and errors

Dashboarding + Analysis

MariaDB
Operational DB

manual code execution to refresh data

Engineering with Python
2020 Q1 data lake start

Google Analytics
www.docsity.com

Google Big Query

G360 data export

Dashboarding + Analysis

MariaDB
Operational DB

manual code execution to refresh data
Automate BigQuery tables from Cloud Storage

Pandas + Google Cloud Storage API

Google Cloud Storage

Google BigQuery API
pip install gcloud-connectors

from gcloud_connectors.gstorage import GStorageConnector

gstorage_service = GStorageConnector(conf_path=None)

... load df

gstorage_service.pd_to_gstorage(
    df=df, bucket_name=bucket_name)
pip install gcloud-connectors

```python
import os
from gcloud_connectors.bigquery import BigQueryConnector

project_id = os.environ['PROJECT_ID']
bq_service = BigQueryConnector(project_id=project_id)
```
Extended gcloud-connectors
2020 Q2 data lake enhancement

Google Analytics
www.docsity.com

G360 data export

Google Big Query

ETL on request

drive to insight on request

MariaDB
Operational DB

code execution to refresh data on request

Huge analytics empowerment

Dashboarding + Analysis
resource "google_bigquery_table" "db_docsity_table" {
  dataset_id = google_bigquery_dataset.db_dataset.friendly_name
  table_id = "db_docsity"
  schema = <<EOF
  [ { "name": "id", "type": "FLOAT", "mode": "NULLABLE" } ... ]
EOF
}
2020 Q2 data lake enhancement

Google Analytics

Lack of orchestration, time spent for manual refresh

MariaDB

Operational DB

code execution to refresh data on request

G360 data export

ETL on request

drive to insight on request

Dashboarding + Analysis

www.docsity.com
2020 Q2 data lake orchestration

This is why Airflow matters
AIRFLOW SECURITY
Security for Google Compute Engine

Compute Engine

service account key & predefined role with granular permissions managed by Terraform

IAM & Admin

Apache Airflow

APIs:
- Google Analytics
- Google Cloud Storage
- Google Sheets
- Google Search Console
- Google Drive
- Google BigQuery
**Airflow encrypted variables & connections**

Encrypted credentials pulled from Compute Engine metadata

<table>
<thead>
<tr>
<th>Key</th>
<th>Val</th>
<th>Is Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD_COMMIT_ID</td>
<td>42354yigdlesfhi83fet3l0t</td>
<td>True</td>
</tr>
<tr>
<td>ENV</td>
<td>************</td>
<td>True</td>
</tr>
<tr>
<td>MARIADB_PASSWORD_CRED</td>
<td>************</td>
<td>True</td>
</tr>
<tr>
<td>MONGODB_PASSWORD_CRED</td>
<td>************</td>
<td>True</td>
</tr>
</tbody>
</table>
**Airflow + Google Secret Manager**

Credentials pulled from Google Secret Manager

---

### List Variable

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Is Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD_COMMIT_ID</td>
<td>42354ywigdlshfhl9klt3olit</td>
<td>True</td>
</tr>
<tr>
<td>ENV</td>
<td>PROD</td>
<td>True</td>
</tr>
</tbody>
</table>
WHAT TIME TO UPDATE?
Schedule time?

AWS RDS prod DB → snapshot to sync → AWS Lambda → AWS RDS data DB

right time to pull data?

Apache Airflow
airflow-add-ons

AWS RDS prod DB

snapshot to sync

AWS Lambda

AWS RDS data DB

semaphore file written to understand when to pull data

AWS S3

class ReturnS3KeySensor
pip install airflow-add-ons

INSTALL ME
https://pypi.org/project/airflow-add-ons/
2020 Q3 data lake orchestration

Google Analytics

www.docsity.com

G360 data export

Google

Big Query

Looker

Dashboarding + Analysis

drive to insight

Pandas ETL

MariaDB

Operational DB

automation + best time to refresh

Apache Airflow

48
pip install pymongo-ssh

dry

```python
from pymongo_ssh import MongoSession

session = MongoSession(
    host='db.example.com',
    port='21',
    user='myuser',
    key='/home/myplace/ssh/id_rsa',
    to_port='37017',
    to_host='0.0.0.0'
)
```
2020 Q4 data lake orchestration

Google Analytics

Google Big Query

Looker
Dashboarding + Analysis

g360 data export

drive to insight

mongoDB
log/review data

MariaDB
Operational DB

automation + best time to refresh

www.docsity.com
2021 Q2 data lake Orchestration

- Google Analytics
  - G360 data export
  - www.docsity.com

- Google Big Query
  - CRUX report
  - automation + best time to refresh

- MongoDB
  - log/review data

- MariaDB
  - Operational DB

- Looker
  - Dashboarding + Analysis

- unbounce
  - Campaign data
  - Google Ad Manager

- Apache Airflow
PRE Airflow

Simple
Errors
Time Spent
Lack of Control

Dashboarding + Analysis

Manual code execution to refresh data

Manual refresh

MariaDB
Operational DB
Thanks!

Does anyone have questions?

matteo.s@docsity.com
https://github.com/pualien
irene.s@docsity.com
docsity.com