Airflow: The Power of stitching Services Together
based on experience with Google Composer

knapik@google.com
rafalbiegacz@google.com
Airflow Summit, July 2021
Filip Knapik

- Cloud Composer Product Manager at Google
- Working with Airflow for 2 years
- 18+ years of IT management experience
- Graduate of Computer Networks and Services at AGH University of Science and Technology in Cracow, Poland
Rafal Biegacz

- **Cloud Composer** Engineering Manager
- Has been working on Airflow for 2 years.
- Holds MSc degree in the field of Teleinformatics from Gdansk University of Technology
- Delivers Google Cloud Platform and cloud computing lectures to students of University of Warsaw and Technical University of Warsaw.
Apache Airflow and Cloud Composer

100% open source compatible

Easy deployment
Managed infrastructure
PyPi packages management
Technical Support
Managed Airflow Environments
Stitching many GCP services to provide managed Airflow environments so you can focus on Airflow and DAG development/execution
Stitched to provide high Security

- IAM
- Encryption
- Perimeter
- Policies
- Infrastructure

Authorized Users

- minimize risk of data exfiltration

Keys managed by users or Google

Org-level Governance

updated by Google

Keys managed by users or Google
Stitched to provide Compliance

- Data Residency
- Assured Workloads
- PCI DSS
- Health industry
- No vendor lock-in

Audit

Location

PCI DSS

Exit Strategy

HPAA

Audited Logs

Access Transparency
Cloud Composer - Monitoring

Monitoring out of the box
Cloud Composer Logging

Logging out of the box

Environment details

MONITORING LOGS PREVIEW

Environment configuration
Airflow configuration overrides
Environment variables
Labels
PyPI packages

6 HOURS

All logs

Airflow logs

Scheduler
Workers
Web server
DAG processor manager
Composer logs
Composer agent
Build
Worker & Scheduler Image
Web server Image
Database operations
Monitoring
Infrastructure
SQL proxy
Redis
GCS syncd
Other

Logs

Severity
Filter
Filter logs


2021-07-04 22:41:31.452 CEST airflow-scheduler DAGs # Errors Last Runtime Last Run

2021-07-04 22:42:42.535 CEST airflow-scheduler INFO - Finding 'running' jobs without a recent heartbeat

2021-07-04 22:42:42.536 CEST airflow-scheduler DAG_PROCESSOR_MANAGER_INFO - Finding 'running' jobs without a recent heartbeat


Stitching Services Together
Stitching - Airflow’s magic power

Build pipelines embracing different services is one of the biggest magic powers of Apache Airflow!

● Connecting to each other totally independent services
● End-2-end observability
● Symbiosis with other workflow technologies !!!
Stitching Enablers ...

Out-of-the-box library of Operators, Hooks and Sensors

Do-It-Yourself Operators

Containers
Airflow Operators, Hooks and Sensors

Apache Airflow only in its code base has:
- Operators: > 470
- Sensors: > 70
- Hooks: > 160
... from > 50 different providers like Amazon AWS™, Qubole™, Google Cloud Platform™, etc.

Google Operators/Hooks/Sensors...
- Operators: 304 (~65% of all operators)
- Sensors: 22 (> 25% of all sensors)
- Hooks: 50 (> 25% of all hooks)
The power of Airflow Operators

Apache Airflow

Bash Operator

Python Operator

print("Hello!")

GCS to BigQuery Operator

GCS to BigQuery

BigQuery Insert Job Operator

BigQuery Insert Job

BigQuery Insert Job

CreateQueryJob(...)

My_Awesome_DAG

GCS to BigQuery

BigQuery Insert Job

BigQuery Insert Job

Python code

Airflow Provider Packages e.g. Google Cloud
Custom Operators

1. Code

```python
from airflow.models.baseoperator import BaseOperator

class HelloOperator(BaseOperator):
    def __init__(self, name: str = None, **kwargs):
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

2. Upload to Airflow

```
from hello_operator import HelloOperator
```

3. Import in your DAG
What if I need special OS-level binaries?

1. Turn into a container

2. Build & push to a container repository (e.g. Google Artifact Registry)
   
   https://[region]-docker.pkg.dev/[project]/[image]

3a. Use KubernetesPodOperator

3b. Use GKEPodOperator
Example Airflow Use Cases
Information about other services used with Cloud Composer (rafal)

BigQuery,

Dataflow, Datafusion, CloudSQL,

// Graphically
SreeTree Use case

Real Use Case: Take pictures of trees and send recommendations to the farmers

Example Composer DAG:
- **Storage**: Collecting the data
- **Dataflow**: Processing and moving the data
- **ML**: Choosing and applying the right AI model
- **BigQuery**: Write the outcome back
Automating runs of Data Fusion pipelines

Event-based triggering

Source: https://cloud.google.com/blog/products/data-analytics/easier-management-for-cloud-elt-elt-pipelines
Loading & enriching data from a transactional system (Filip)

Retailer triggering Airflow DAG to:
1. Load data from Salesforce to BigQuery at a scheduled interval
2. Process the data in BigQuery using SQL queries (ELT)
3. Generate data marts that users access using BI tools

Orchestrated by Airflow / Cloud Composer

Change to CRM : Salesforce
Machine Learning training

ML Training

1. BigQuery
2. Cloud Storage
3. GKE (GPU/TPU)
4. Cloud Storage

Orchestrated by Airflow / Cloud Composer (weekly)

Predictions
Big Data Spark jobs in ephemeral clusters

Large operator of a marketplace service:
1. Trigger a DAG when the file arrives
2. Create a Dataproc cluster
3. Run a job and push its result to GCS
4. Delete a Dataproc cluster (ephemeral!)

Orchestrated by Airflow / Cloud Composer
CloudML Use Case
Summary
Why Airflow?

Large operator of a marketplace service:
1. Rich integrations
2. Extensibility with own operators
3. Ability to schedule custom non-python tasks
4. Stitching services with Airflow provides
   a. Observability
   b. Easier troubleshooting
   c. Simpler change management
   d. Integrated security
Thank You