Airflow: The Power of stitching Services Together

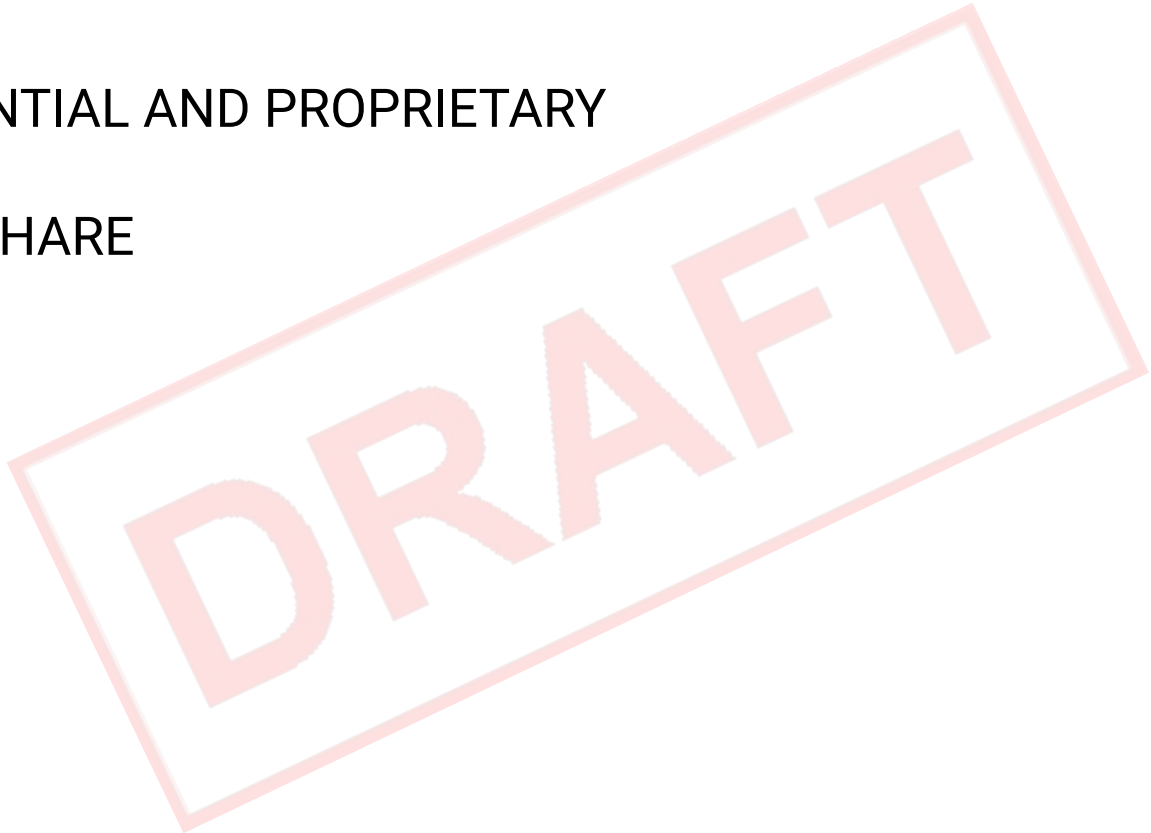
based on experience with Google Composer

knapik@google.com
rafalbiegacz@google.com

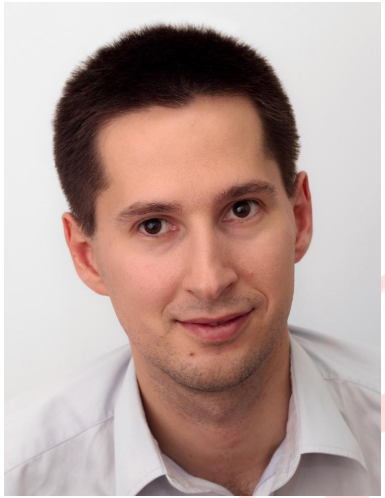
Airflow Summit, July 2021

CONFIDENTIAL AND PROPRIETARY

DO NOT SHARE



BIO



Filip Knapik

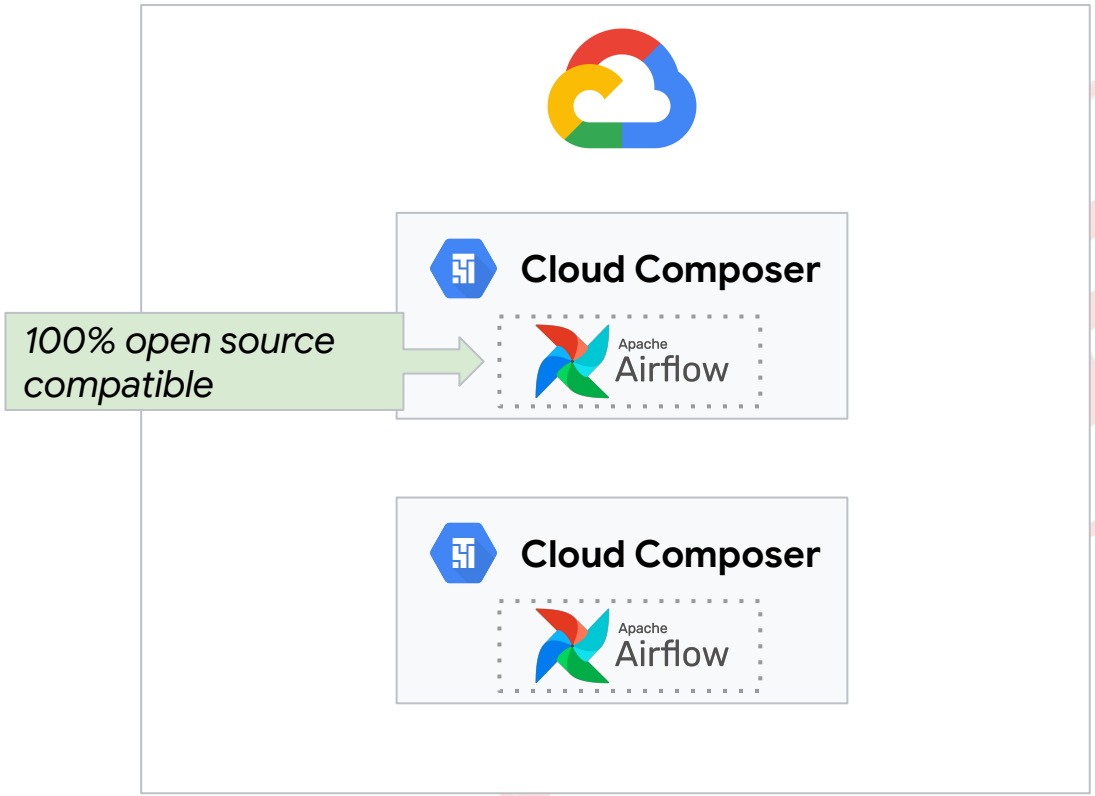
- Cloud Composer Product Manager at Google
- Working with Airflow for 2 years
- 18+ years of IT management experience
- Graduate of Computer Networks and Services at AGH University of Science and Technology in Cracow, Poland

BIO

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



Easy deployment

Managed infrastructure

PyPi packages management

Technical Support

Managed Airflow Environments

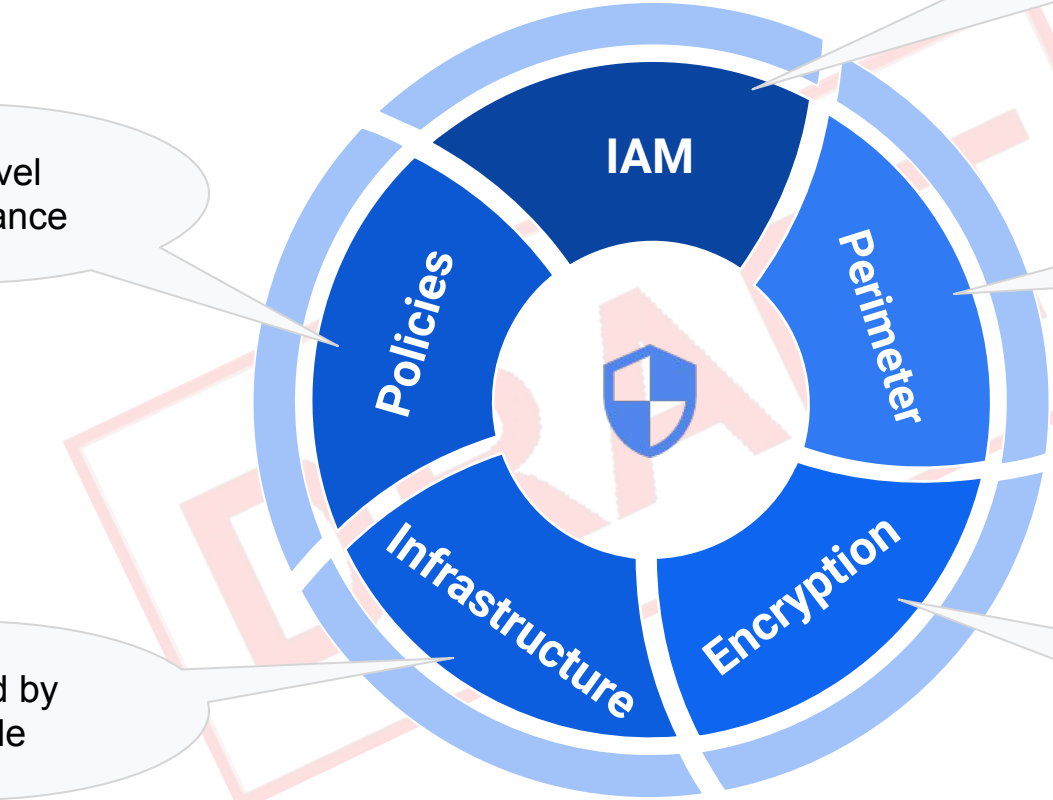
Stitching many GCP services to provide

managed Airflow environments

so you can focus on

Airflow and DAG development/execution

Stitched to provide high Security



Authorized Users

minimize risk of data exfiltration

Keys managed by users or Google

updated by Google

Org-level Governance

Stitched to provide Compliance

Audited Logs
Access Transparency

Health industry

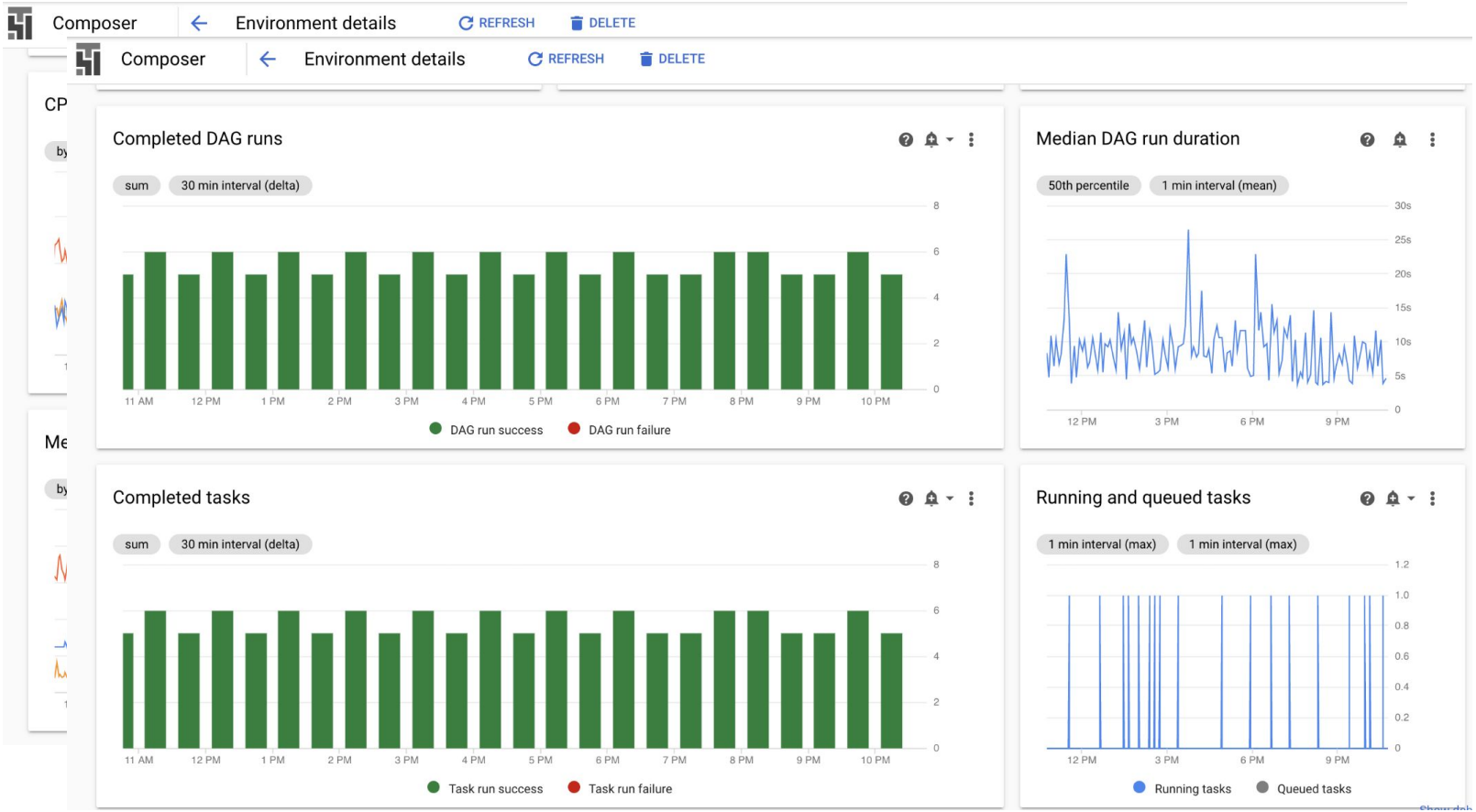
No vendor lock-in

Finance

Data Residency
Assured Workloads

Cloud Composer - Monitoring

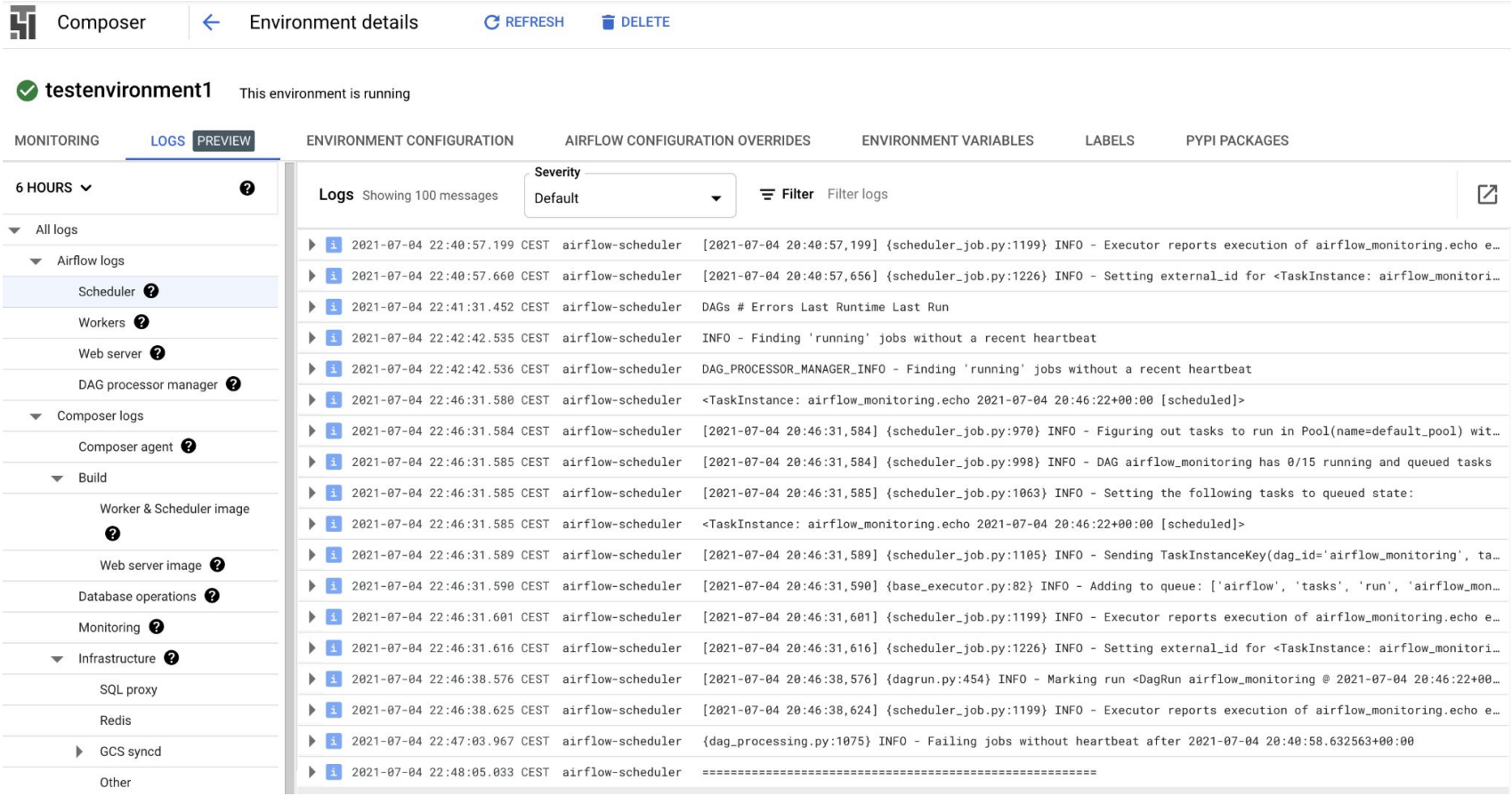
Monitoring out of the box



Google

Cloud Composer Logging

Logging out of the box



Google

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

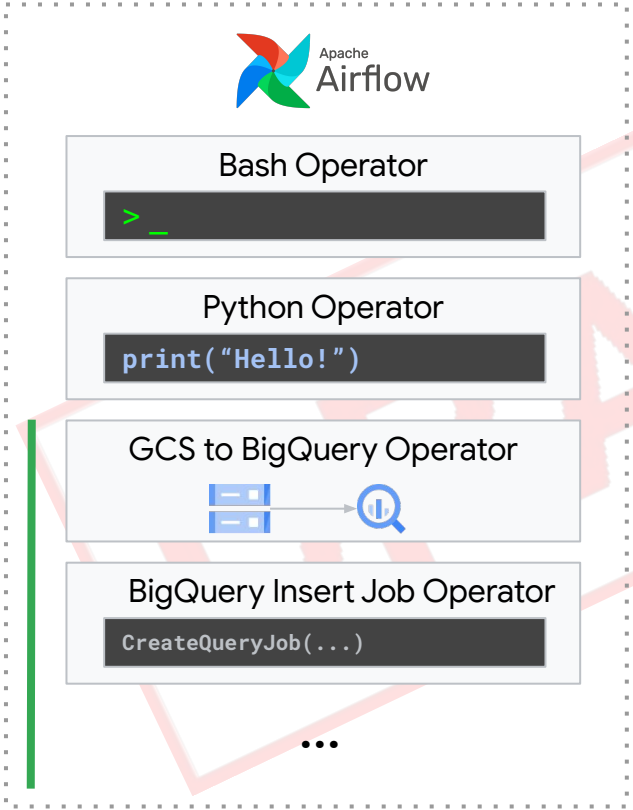
… from > 50 different providers like Amazon AWS™, Qubole™ , Google Cloud Platform™ , etc.

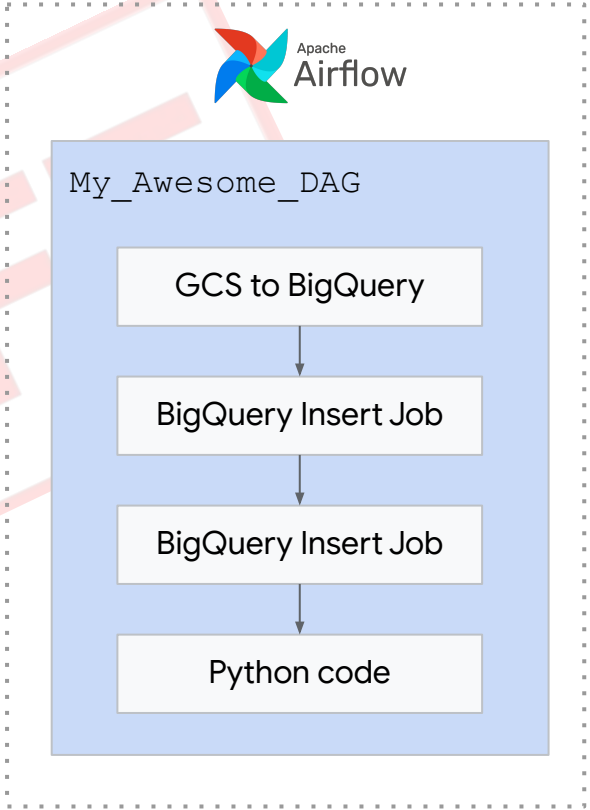
Google Operators/Hooks/Sensors...

- Operators: 304 (~65% of all operators)
- Sensors: 22 (> 25% of all sensors)
- Hooks: 50 (> 25% of all hooks)

The power of Airflow Operators

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,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

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

2. Upload to Airflow

Save `hello_operator.py` in `/plugins/` folder

3. Import in your DAG

```python
from hello_operator import HelloOperator
```

What if I need special OS-level binaries?

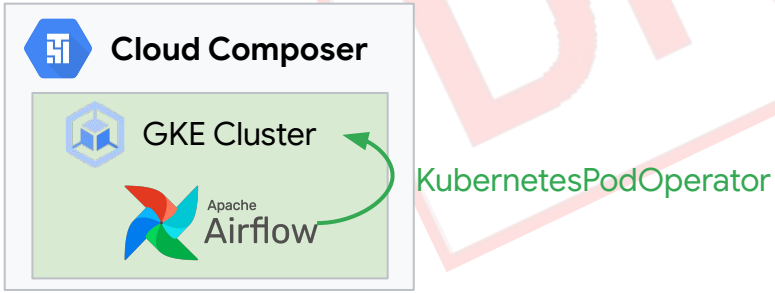
1. Turn into a container

```
FROM python:3.8-slim

ENV APP_HOME /app
WORKDIR $APP_HOME

RUN apt -y update
RUN apt -y install gcc
...
```

2. Build & push to a container repository
(e.g. Google Artifact Registry)

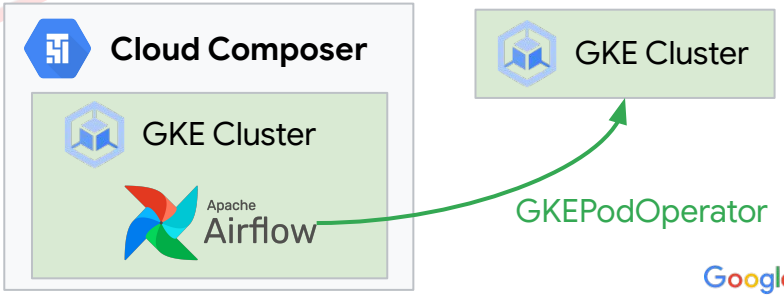
https://[region]-docker.pkg.dev/[project]/[image]

3a. Use KubernetesPodOperator



OR

3b. Use GKEPodOperator



Example Airflow Use Cases

Information about other services used with Cloud Composer (rafal)

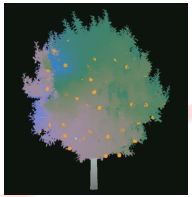
BigQuery,

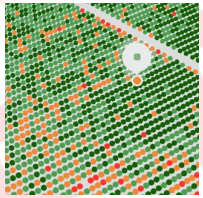
Dataflow, Datafusion, CloudSQL,

// Graphically

SreeTree Use case

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





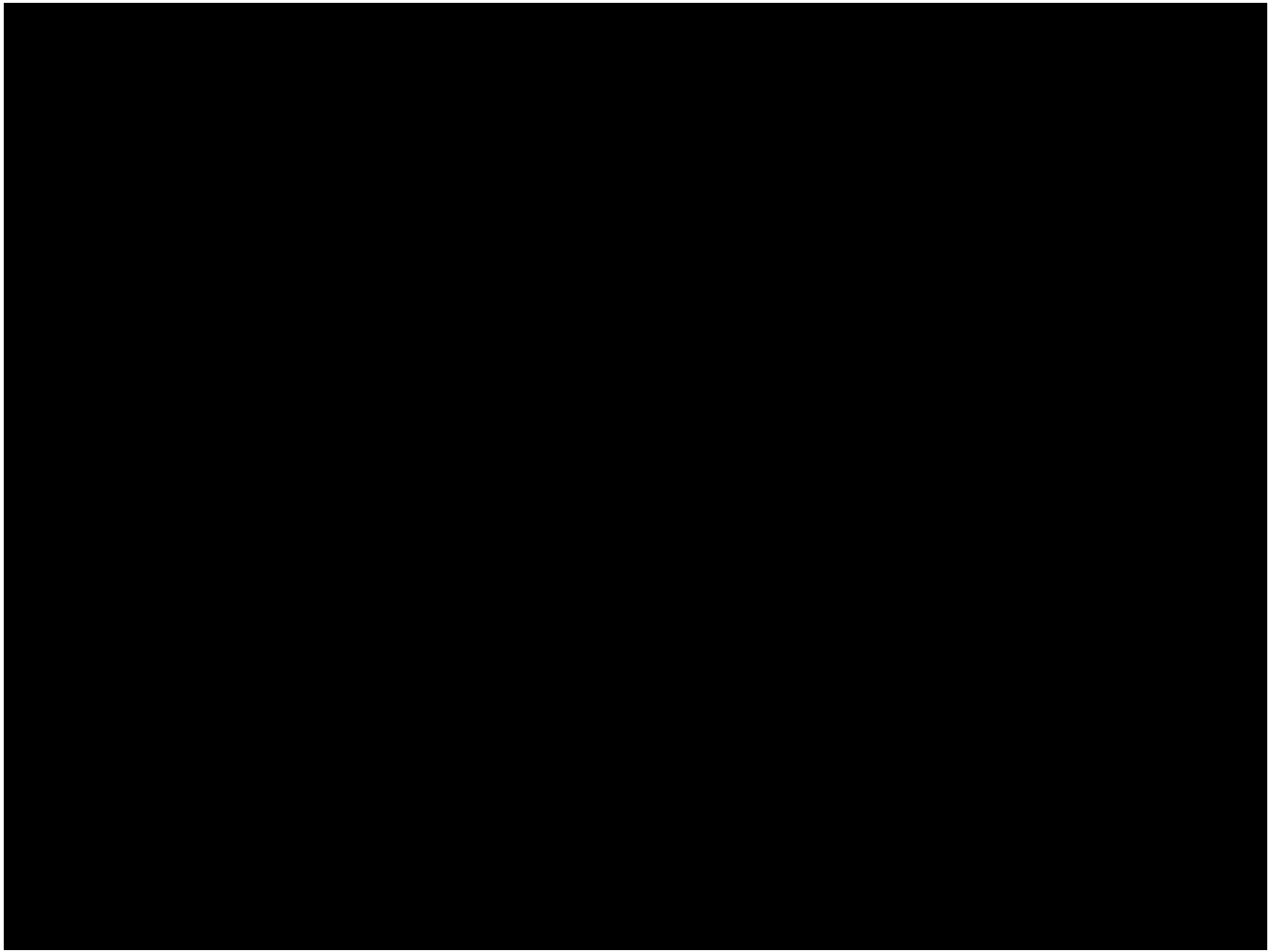
Example Composer DAG

Collecting the data

Processing and moving the data

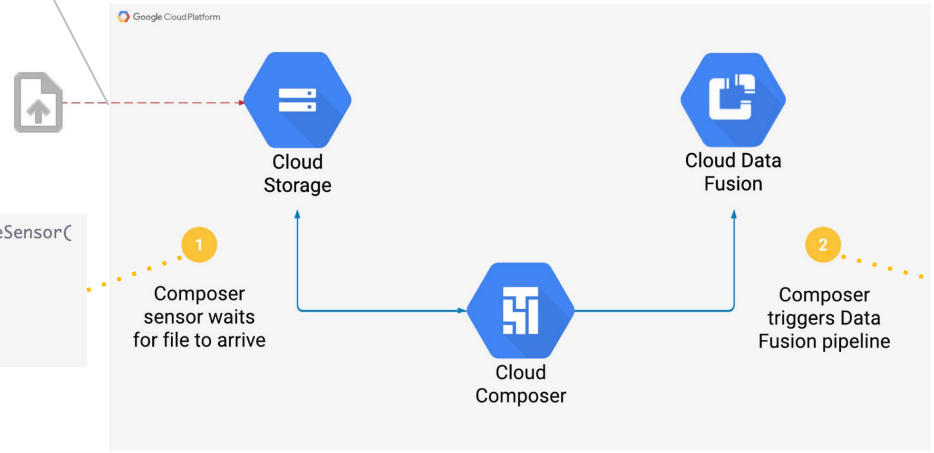
Choosing and applying the right AI model

Write the outcome back



Automating runs of Data Fusion pipelines

Event-based triggering




```
gcs_sensor_task = GCSObjectExistenceSensor(
    task_id="gcs_object_sensor",
    bucket='my_data_bucket',
    object='my_data_01082020.csv',
    dag=dag
)
```

```
start_pipeline_task = CloudDataFusionStartPipelineOperator(
    task_id="start_cdf_pipeline",
    location='us-west1',
    pipeline_name ='demo_pipeline',
    instance_name="demo_instance",
    runtime_args={'input_dir':'my_data_01082020.csv'},
    dag=dag
)
```

```
gcs_sensor_task >> start_pipeline_task
```

Source: https://cloud.google.com/blog/products/data-analytics/easier-management-for-cloud-etl-elt-pipelines

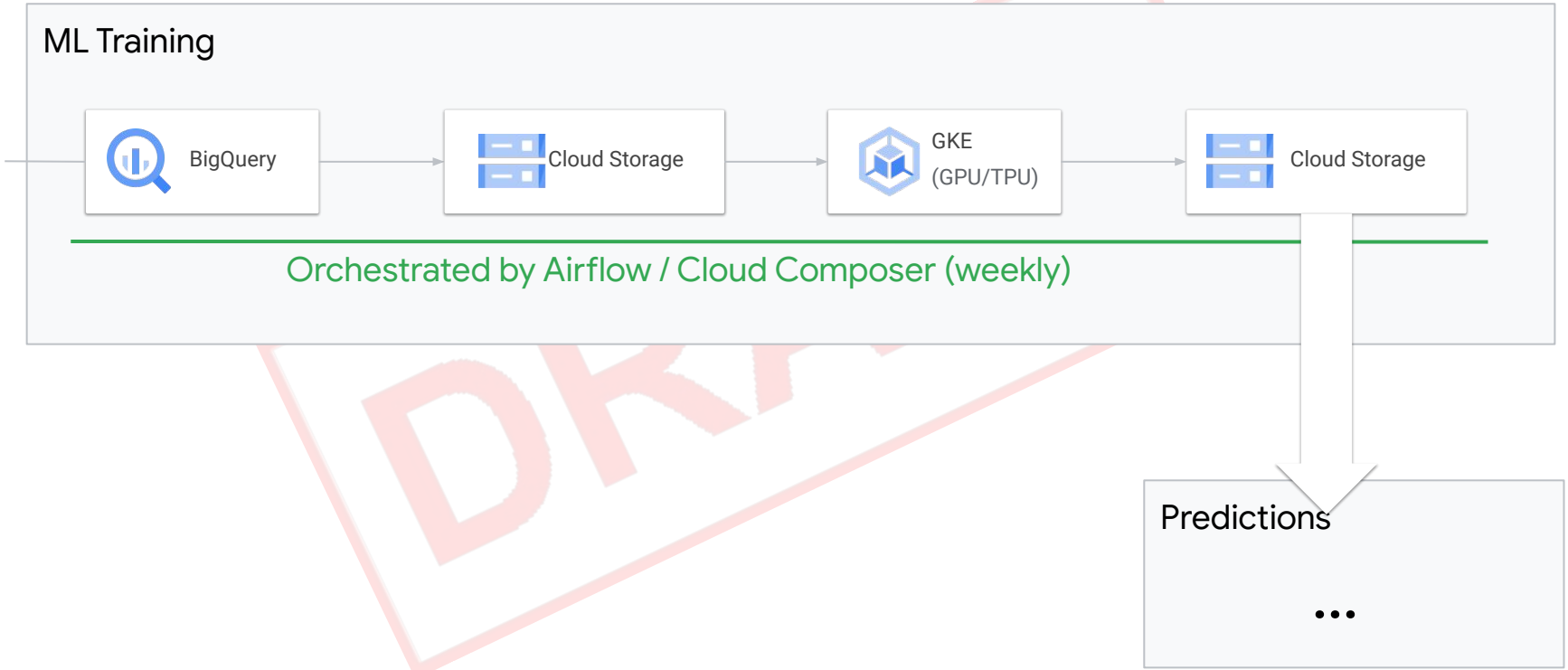
Loading & enriching data from a transactional system (Filip)

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

Change to CRM : Salesforce

Machine Learning training



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!)

CloudML Use Case

<format of all slides should be equal>

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