Reuniting the two distant cousins: *Calling a Beam Pipeline from an Airflow Job*

Sadeeq Akintola

X / Twitter: @SadeeqAkintola
Online: SadeeqAkintola.com

BEAM SUMMIT

September 4-5, 2024

Sunnyvale, CA. USA

About the Speaker

Sadeeq Akintola

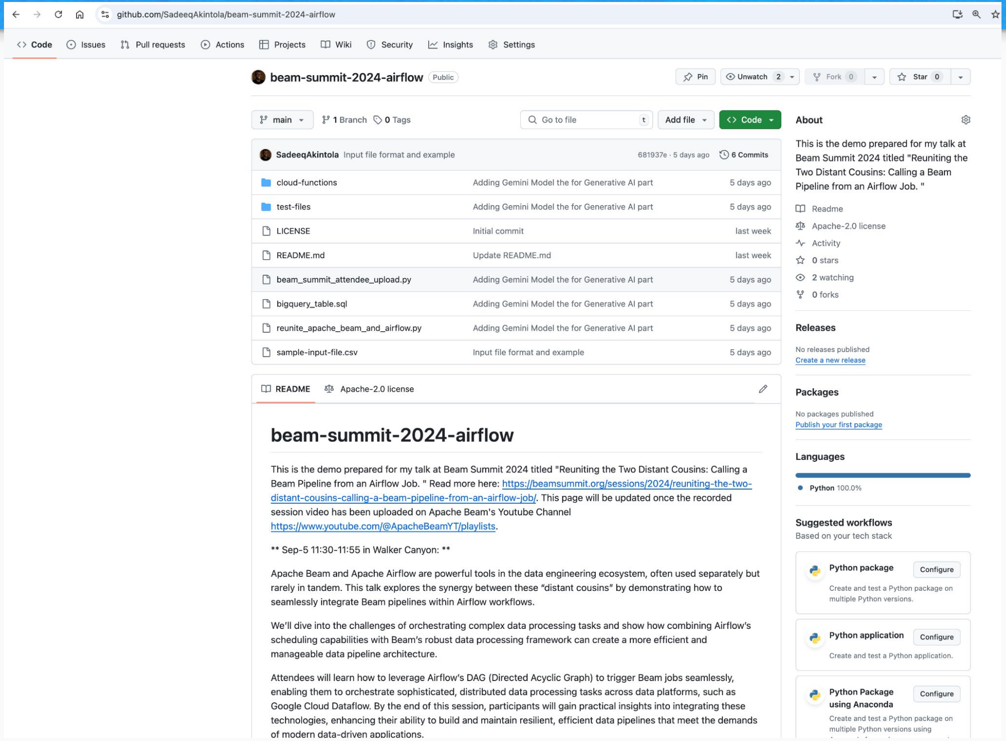
- Customer Engineer, Data Analytics Specialist @ **Google** Cloud
- 13+ Years Industry Experience
- Previous job roles include: Software Engineer, Data Scientist & BI Analyst, Big Data Engineer
- Ex-Microsoft, ex-KPMG, ex-FMDQ Exchange
- Worked across multiple Geos: *Nigeria → Portugal → United Kingdom*
- M.Sc. Data Science and Advanced Analytics @NovaIMS, Lisbon.

Google Cloud Certified PROFESSIONAL DATA ENGINEER

BEAM SUMMIT

Agenda

Try the Demo, Clone the Repo.



SadeeqAkintola Input file format and example 681937e · 5 days ago 6 Commits

cloud-functions	Adding Gemini Model the for Generative AI part	5 days ago
test-files	Adding Gemini Model the for Generative AI part	5 days ago
LICENSE	Initial commit	last week
README.md	Update README.md	last week
beam_summit_attendee_upload.py	Adding Gemini Model the for Generative AI part	5 days ago
bigquery_table.sql	Adding Gemini Model the for Generative AI part	5 days ago
reunite_apache_beam_and_airflow.py	Adding Gemini Model the for Generative AI part	5 days ago
sample-input-file.csv	Input file format and example	5 days ago

README Apache-2.0 license

beam-summit-2024-airflow

This is the demo prepared for my talk at Beam Summit 2024 titled "Reuniting the Two Distant Cousins: Calling a Beam Pipeline from an Airflow Job. " Read more here: https://beamsummit.org/sessions/2024/reuniting-the-two-distant-cousins-calling-a-beam-pipeline-from-an-airflow-job/. This page will be updated once the recorded session video has been uploaded on Apache Beam's Youtube Channel https://www.youtube.com/@ApacheBeamYT/playlists.

** Sep-5 11:30-11:55 in Walker Canyon: **

Apache Beam and Apache Airflow are powerful tools in the data engineering ecosystem, often used separately but rarely in tandem. This talk explores the synergy between these "distant cousins" by demonstrating how to seamlessly integrate Beam pipelines within Airflow workflows.

We'll dive into the challenges of orchestrating complex data processing tasks and show how combining Airflow's scheduling capabilities with Beam's robust data processing framework can create a more efficient and manageable data pipeline architecture.

Attendees will learn how to leverage Airflow's DAG (Directed Acyclic Graph) to trigger Beam jobs seamlessly, enabling them to orchestrate sophisticated, distributed data processing tasks across data platforms, such as Google Cloud Dataflow. By the end of this session, participants will gain practical insights into integrating these technologies, enhancing their ability to build and maintain resilient, efficient data pipelines that meet the demands of modern data-driven applications.

About

This is the demo prepared for my talk at Beam Summit 2024 titled "Reuniting the Two Distant Cousins: Calling a Beam Pipeline from an Airflow Job. "

Readme
Apache-2.0 license
Activity
0 stars
2 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

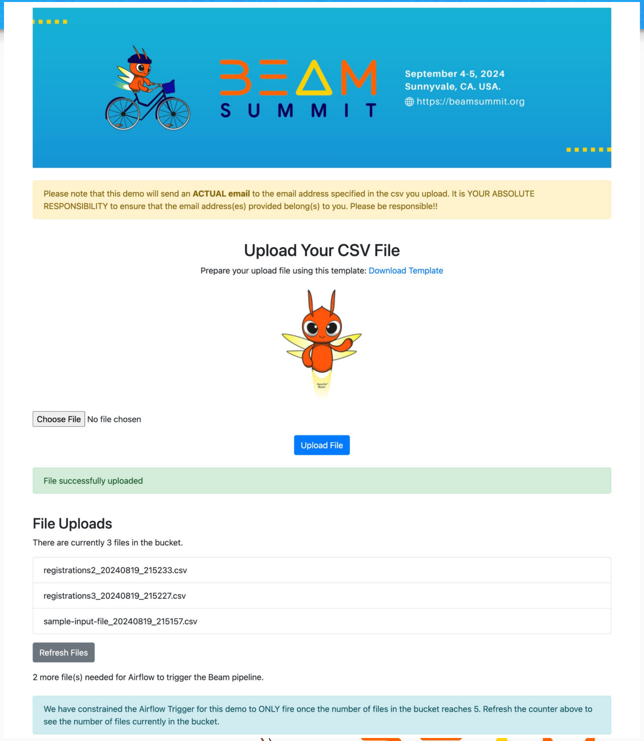
Python 100.0%

Suggested workflows
Based on your tech stack

Python package — Configure
Create and test a Python package on multiple Python versions.

Python application — Configure
Create and test a Python application.

Python Package using Anaconda — Configure
Create and test a Python package on multiple Python versions using



Please note that this demo will send an **ACTUAL email** to the email address specified in the csv you upload. It is YOUR ABSOLUTE RESPONSIBILITY to ensure that the email address(es) provided belong(s) to you. Please be responsible!!

Upload Your CSV File

Prepare your upload file using this template: Download Template

Choose File No file chosen

Upload File

File successfully uploaded

File Uploads

There are currently 3 files in the bucket.

registrations2_20240819_215233.csv
registrations3_20240819_215227.csv
sample-input-file_20240819_215157.csv

Refresh Files

2 more file(s) needed for Airflow to trigger the Beam pipeline.

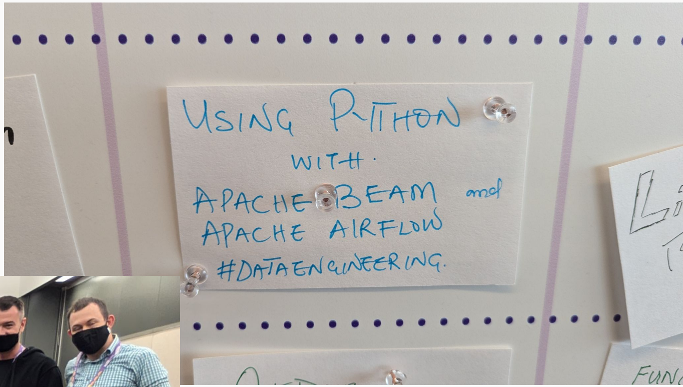
We have constrained the Airflow Trigger for this demo to ONLY fire once the number of files in the bucket reaches 5. Refresh the counter above to see the number of files currently in the bucket.

Visit http://35.222.154.240:8080/ or https://SadeeqAkintola.com/ to try the demo.
Please Star, Fork and Clone the Code Repo here: https://github.com/SadeeqAkintola/beam-summit-2024-airflow

Motivation for this topic:

PyCon US·24
OPEN SPACES SCHEDULE
If you would like to add a gathering, please fill out an index card and pin it in an available room and time
Want to see the board on-the-go?
You can see the open spaces board on your phone! See this board at http://pycon.us/os
SATURDAY, MAY 18
Room 308 Capacity: 60
Room 309 Capacity: 60
Room 315 Capacity: 80
Room 316 Capacity: 80
Room 318 Capacity: 80
Room 320 Capacity: 80
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
5PM
6PM
7PM
8PM

- Previous Experience as a Data Analytics Consultant
- Current Customer Interactions @ Google
- PyCon US Open Spaces Session
- People always love the idea of a Magic Wand product that can do everything, perfectly :)



Understanding Important Concepts

The Data Pipeline Platform

There are certain (other) requirements expected of a Modern Data Pipeline Platform:

Schedule	Dependencies	Error Handling	Reporting	Connectors

Three Common Types of Data Pipelines

Realtime

Realtime (or near realtime) data. An unbounded stream of data that it is desirable to process within a short period of time (mins → microseconds).

Transport Example: Traffic sensors sending information about traffic flow. This may require quick action to be taken (e.g. open / close lanes)

Scheduled Batch

A bounded set of data which requires processing at regular intervals (e.g. every day, 4 hours, 5 minutes)

Retail Example: Looking at purchasing patterns in store yesterday and comparing with online. Maybe looking for online to offline visits.

Triggered Batch

A bounded set of data which requires processing in a reactive manner when an event occurs (e.g. a data file is uploaded to a folder)

Telecoms Example: Processing audio files from support call center as they are uploaded to storage. Running through voice to text conversion then processing for keywords / sentiment.

Data Pipeline Tools

Popular tools on Google Cloud used to Processing each Pipelines:

Realtime

Cloud Pub/Sub

Cloud Dataflow

Scheduled Batch

Cloud Dataflow

Apache Beam

App Engine Scheduled Tasks

Triggered Batch

Cloud Composer

Apache Airflow

Cloud Functions

Note: There are other tools in the open source space such as: Kafka, RabbitMQ, Luigi, Oozie, Azkaban, Hadoop etc. A combination of two or more of these might equally be suitable, depending on the use case.

BEAM SUMMIT

Challenges in Orchestrating Data Processing Tasks

Organizing Data can be such a pain…

- **Complexity of Data Workflows**
  - Managing dependencies and data flow between tasks.
  - Ensuring data consistency and integrity across distributed systems.

- **Scalability Concerns**
  - Handling large volumes of data efficiently.
  - Scaling resources dynamically based on workload.

- **Error Handling and Monitoring**
  - Detecting and recovering from failures.
  - Monitoring pipeline performance and resource utilization.

Orchestration Need: A pipeline orchestrator manages scheduling, monitoring, and dependencies, ensuring smooth data flow.

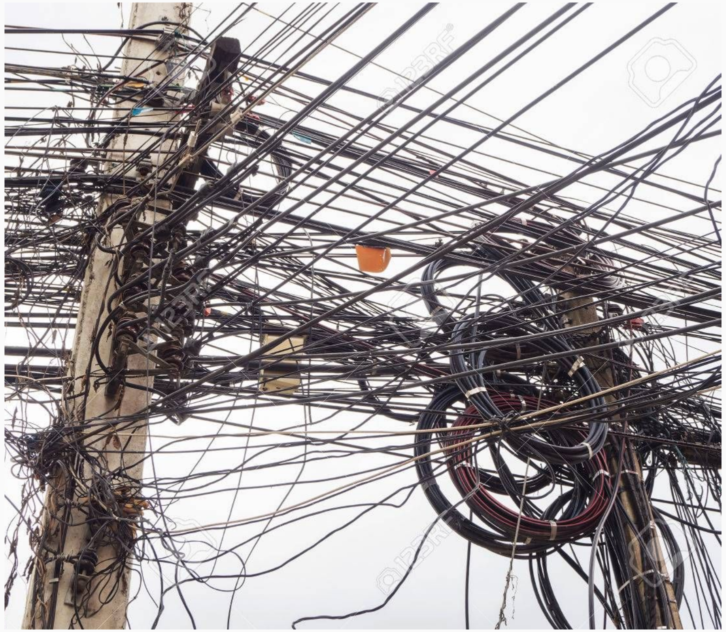


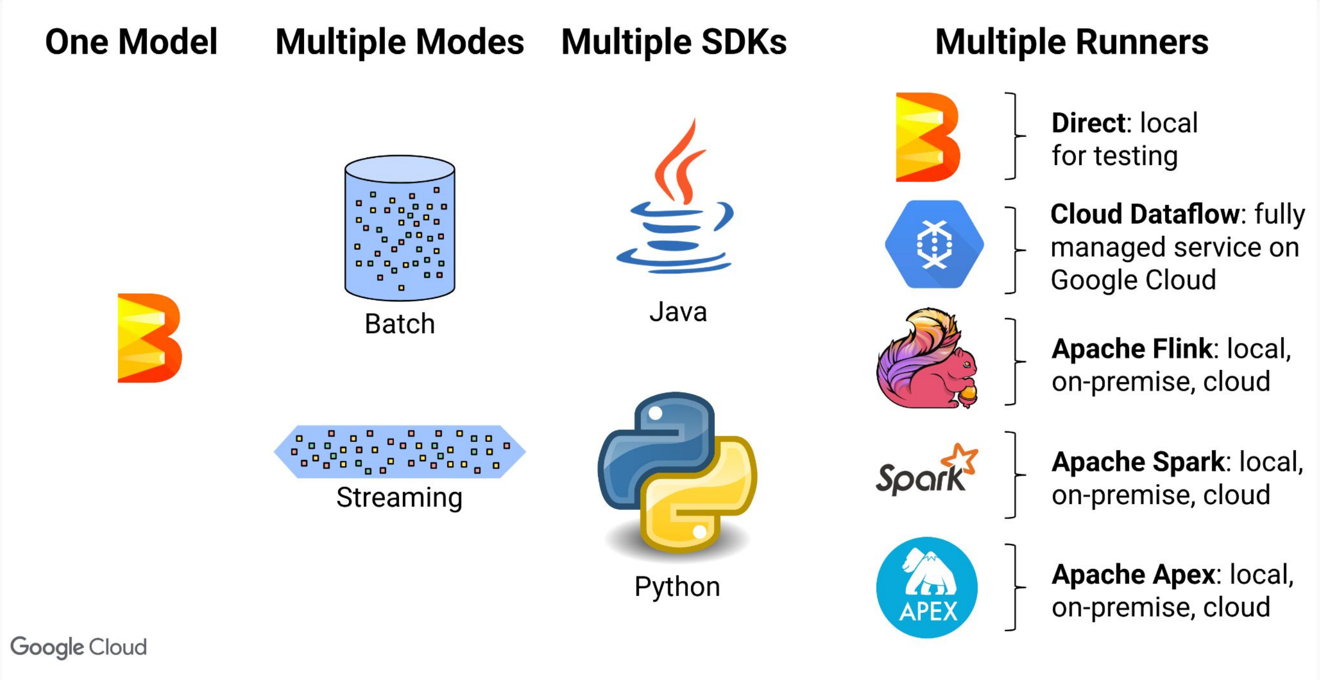
Image source: https://www.123rf.com/photo_65290422_chaos-of-cables-and-wires-on-electric-pole-in-chiang-mai-thailand.html

BEAM SUMMIT

Apache BEAM (Batch + StrEAM)

Apache Beam: Core Concepts

- Open source, unified model for batch and streaming data pipelines
- Using one of the open source SDKs, you can build a program that defines the pipeline
- The pipeline is then executed by one of Beam's supported runners - Apache Apex, Flink, Spark or Google Cloud Dataflow



Direct: local for testing

Cloud Dataflow: fully managed service on Google Cloud

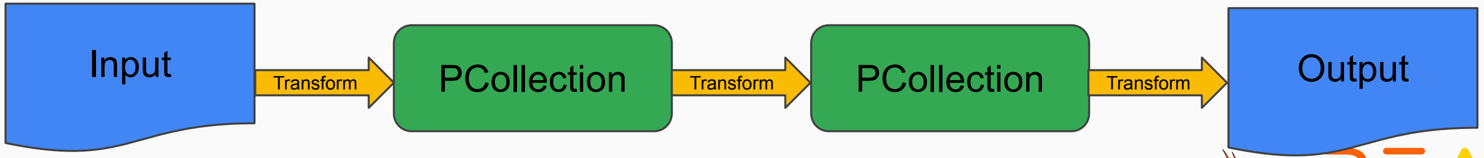
Apache Flink: local, on-premise, cloud

Apache Spark: local, on-premise, cloud

Apache Apex: local, on-premise, cloud

Apache Beam: Core Concepts

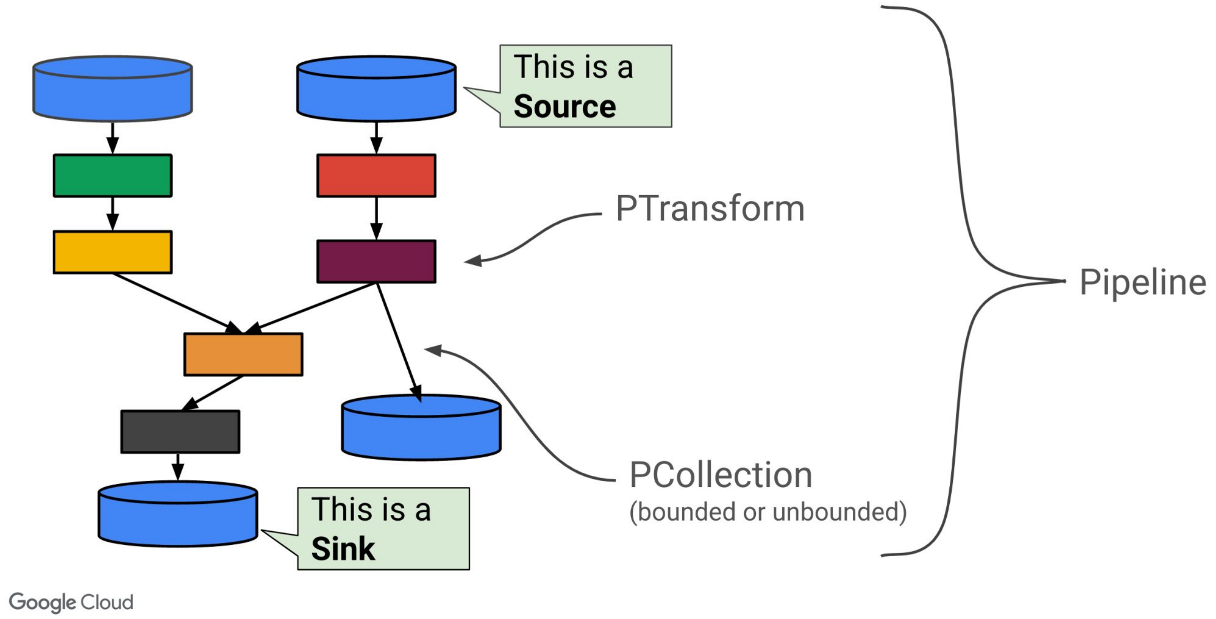
- **Pipeline:** encapsulates the entire series of computations involved in reading input data, transforming that data, and writing output data.
- **PCollection:** represents a potentially distributed, multi-element dataset that acts as the pipeline's data. Beam transforms use PCollection objects as inputs and outputs for each step in your pipeline.
- **PTransforms:** A transform represents a processing operation that transforms data. A transform takes one or more PCollections as input, performs an operation that you specify on each element in that collection, and produces one or more PCollections as output.
- **I/O Sinks and Sources** – The Source and Sink APIs provide functions to read data into and out of collections. The sources act as the roots of the pipeline and the sinks are the endpoints of the pipeline.



*Example pipeline [Apache Beam Programming Guide]

Pipelines

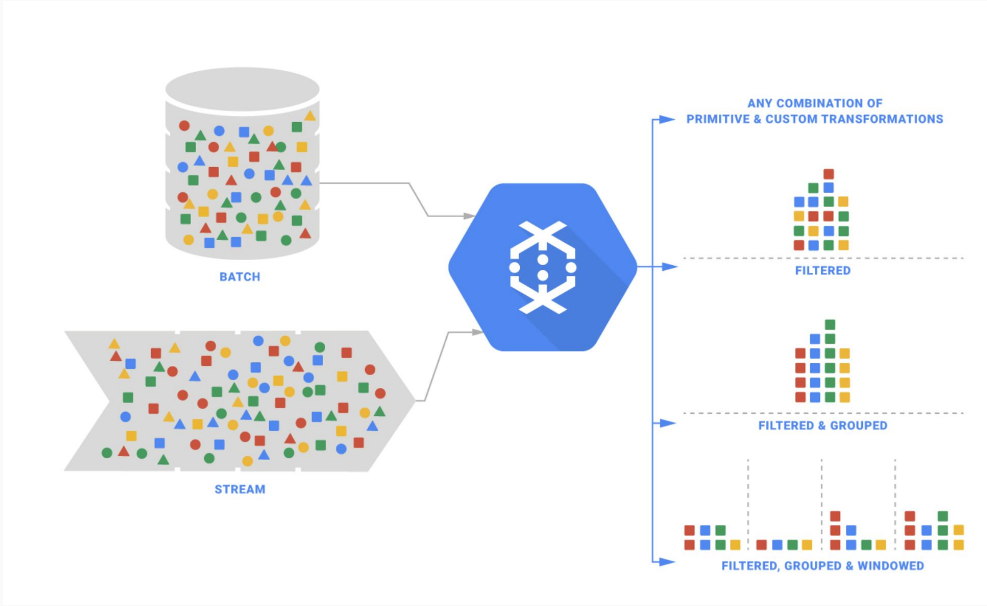
- In Beam, you structure your computation as a graph of transformations, which we call a **Pipeline**.
- Each box here is a transform performing massively parallel computation, which we call a **PTransform**.
- Each Transform of the Pipeline is *applied* on a **PCollection**; the result of apply() is another **PCollection**.
- Each arrow represents the data itself, being transmitted from one PTransform to the next, which we call a PCollection.



PCollection

A PCollection represents a distributed data set that can be **bounded** or **unbounded**:

- Bounded means we know the PCollection is finite, while unbounded means that it ***might be infinite***, ***it might be finite***, but we just don't know.
- A Directed Acyclic Graph of data transformations.
- Possibly unbounded collections of data flow on the edges.
- May include multiple sources and multiple sinks.
- Optimized and executed as a unit.
- PCollections are immutable.


BATCH
STREAM
ANY COMBINATION OF PRIMITIVE & CUSTOM TRANSFORMATIONS
FILTERED
FILTERED & GROUPED
FILTERED, GROUPED & WINDOWED

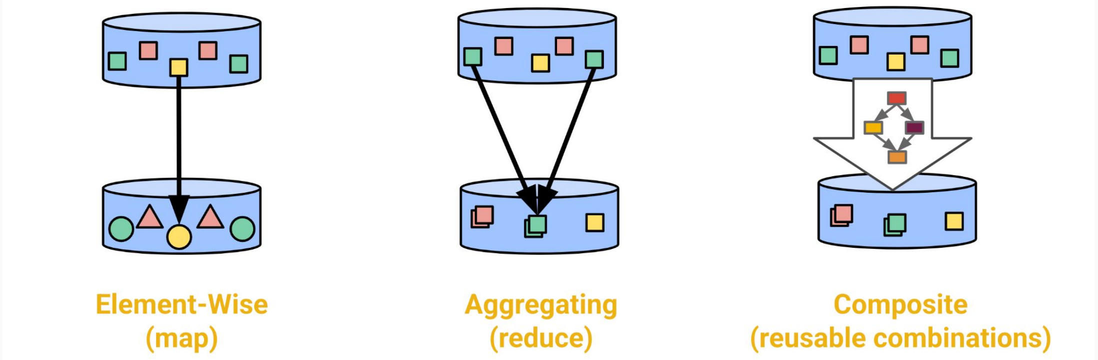
BEAM SUMMIT

PTransforms

- **Element-wise Transformation:** let say you process element individually and do some transformation on it. For example, you have a record with the user id and transform the user id to an email address.
    - Most frequently used *Map Transform* is *ParDo* = “Parallel Do”.

- **Aggregating Transform**, also called *reduce*. Where different elements are processed together.
    - The key primitive is the *GroupByKey* - which groups key-value pairs by key.
    - It takes multiple elements and combines them.

- **Composite transformation:** they’re just compound operations of more primitive things. For example, you can have a *combine fn* that counts words and then extract the top-K elements.

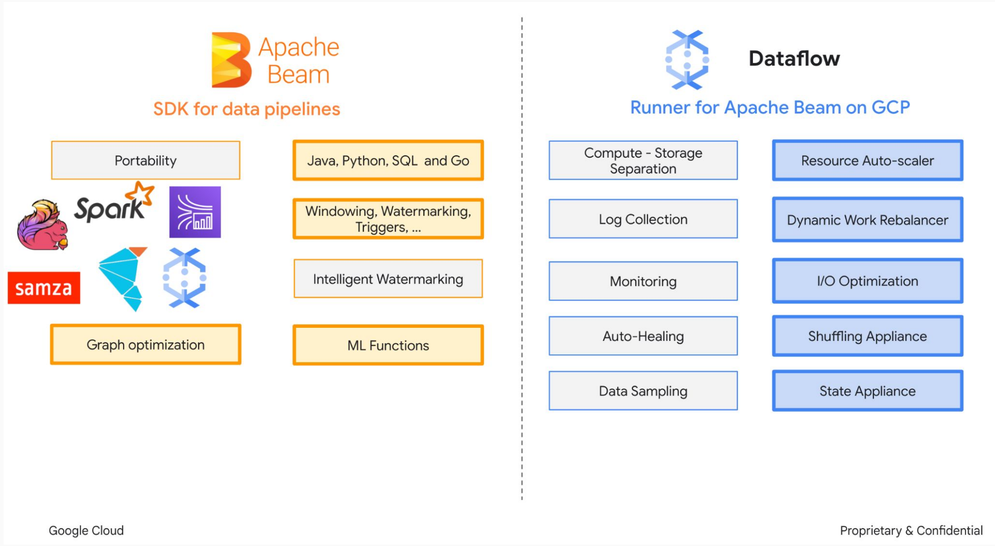


BEAM SUMMIT

Why run Beam on Google Cloud Dataflow?

Cloud native, serverless, extensible solution for mission critical ingestion, ETL, and streaming analytics:

- Fully-managed and auto-configured
  - Resource management: Spinning up and down the machines that process data.
  - Dynamic work rebalancing: Partition and spread the data so that all machines have work to do, all the time.
- Auto graph-optimized for best execution path
- Autoscaling mid-job: if the load goes up or down, adjust the infrastructure accordingly.
- Dynamic Work Rebalancing mid-job



Dataflow: Platform Powered by Google + Rich Open Source Apache Beam SDK

Apache Airflow

Apache Airflow: Core Concepts

- Apache Airflow is an open-source workflow management platform for data engineering pipelines.
- It started at Airbnb in October 2014 as a solution to manage the company's increasingly complex workflows.
- Apache Airflow is used for the scheduling and orchestration of data pipelines or workflows.
- Orchestration of data pipelines refers to the sequencing, coordination, scheduling, and managing of complex data pipelines from diverse sources.

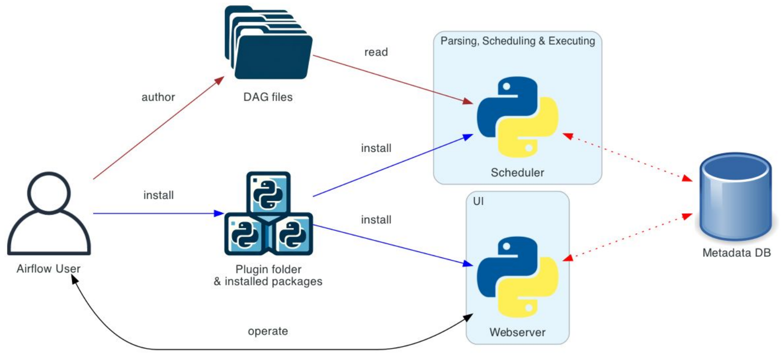


Image source: https://airflow.apache.org/docs/apache-airflow/stable/_images/diagram_basic_airflow_architecture.png

Apache Airflow: Workflow Principles

These are the key principles you need to know about when building an Airflow workflow

- .py File
- DAG
- Arguments
- Schedule
- Tasks / Operators
- Macros
- Dependencies

Each Airflow workflow is a python file that is placed in the dags folder where Airflow runs.

The python syntax used to build the workflow is very simple and makes use of the operators with simple arguments for each task.

Configuration as code instead of drag and drop UI.

Apache Airflow: Core Concepts

- **Directed Acyclic Graphs (DAGs):**
  - Represents workflows as a collection of tasks with defined dependencies.
  - Enables workflows to be defined as code for better maintainability.
- **Tasks**:
  - Basic units of execution within a DAG.
  - Can perform various operations like data fetching and analysis.
- **Operators:**
  - Templates that define what a task does (e.g., BashOperator, PythonOperator).
  - Variety available for different use cases.
- **Hooks**:
  - Interfaces to external platforms and services.
  - Used by operators for tasks like database queries or API calls.
- **XComs:**
  - Mechanism for tasks to exchange small amounts of data.
  - Facilitates communication between tasks in a DAG.
- **Architecture:**
  - Scheduler: Manages task execution and DAG scheduling.
  - Web Server: Provides a UI for monitoring and managing workflows.
  - Metadata Database: Stores the state of tasks and workflows.
  - Executor: Executes tasks locally or on distributed systems.
- **Extensibility and Community:**
  - Highly extensible with custom operators and hooks.
  - Supported by a large, active open-source community.

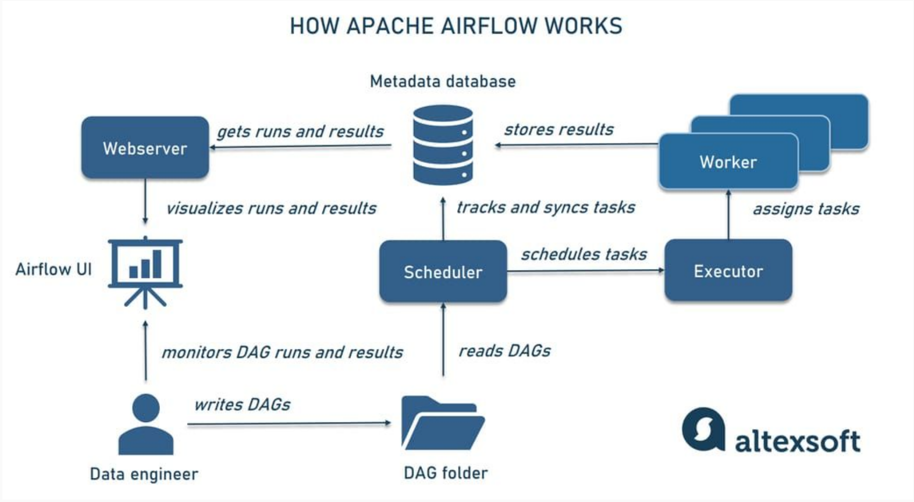


Image source: https://www.altexsoft.com/static/blog-post/2023/11/fc3dd59e-998e-4288-aac1-f172a9204832.jpg

Apache Airflow: Directed Acyclic Graphs (DAGs)

A DAG in Apache Airflow is a central concept that represents a workflow of tasks organized in a way that defines their execution order. Here’s a detailed explanation:

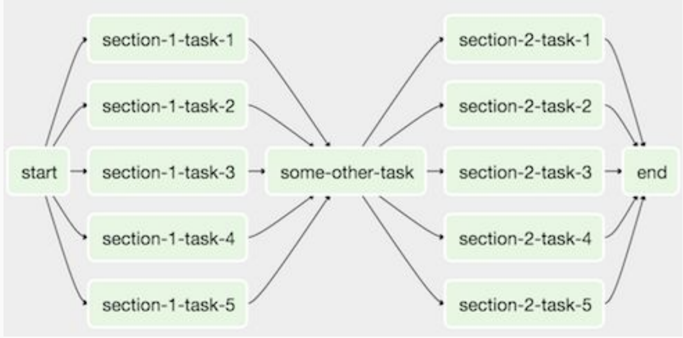
- **Directed**: The connections between tasks have a direction, meaning Task A must complete before Task B starts, establishing a clear sequence of execution.

- **Acyclic**: means there are no loops. Once a task is executed, the workflow doesn’t return to that task; it progresses forward.

- **Graph**:The DAG is essentially a collection of nodes (tasks) and edges (dependencies) that represent the workflow. It visualizes the entire pipeline of tasks from start to finish.

- **Components of a DAG**
    - Dependencies
    - Schedule
    - Operators
    - Execution

```
from airflow import DAG
dag = DAG('my-new-dag', schedule_interval='0 0 * * *')
```



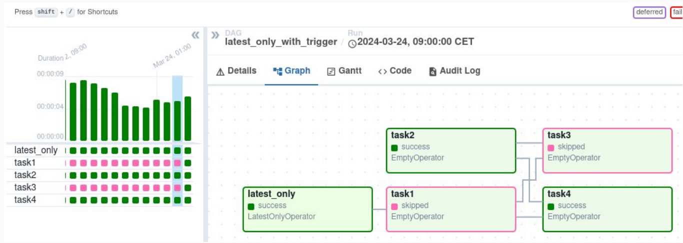

Press shift + / for Shortcuts
deferred
DAG latest_only_with_trigger
Run 2024-03-24, 09:00:00 CET
Details
Graph
Gantt
Code
Audit Log
Duration
00:00:09
00:00:04
00:00:00
latest_only
task1
task2
task3
task4
task2 success EmptyOperator
task3 skipped EmptyOperator
latest_only success LatestOnlyOperator
task1 skipped EmptyOperator
task4 success EmptyOperator

Image source: https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html

Why run Airflow on Google Cloud Composer?

Apache Airflow

+ Python/open source
+ Well-established interfaces
  (CLI / Webserver / API server)
+ Flexible scheduling and dependency management with retries
+ Rich library of connectors
+ Active community support
- Non-trivial setup & management efforts
- Logging/debugging
+ Apache Airflow supports many executors: local, Celery, Kubernetes, Mesos, Dask

benefits from ▶

◀ contributes to

Google Cloud Platform

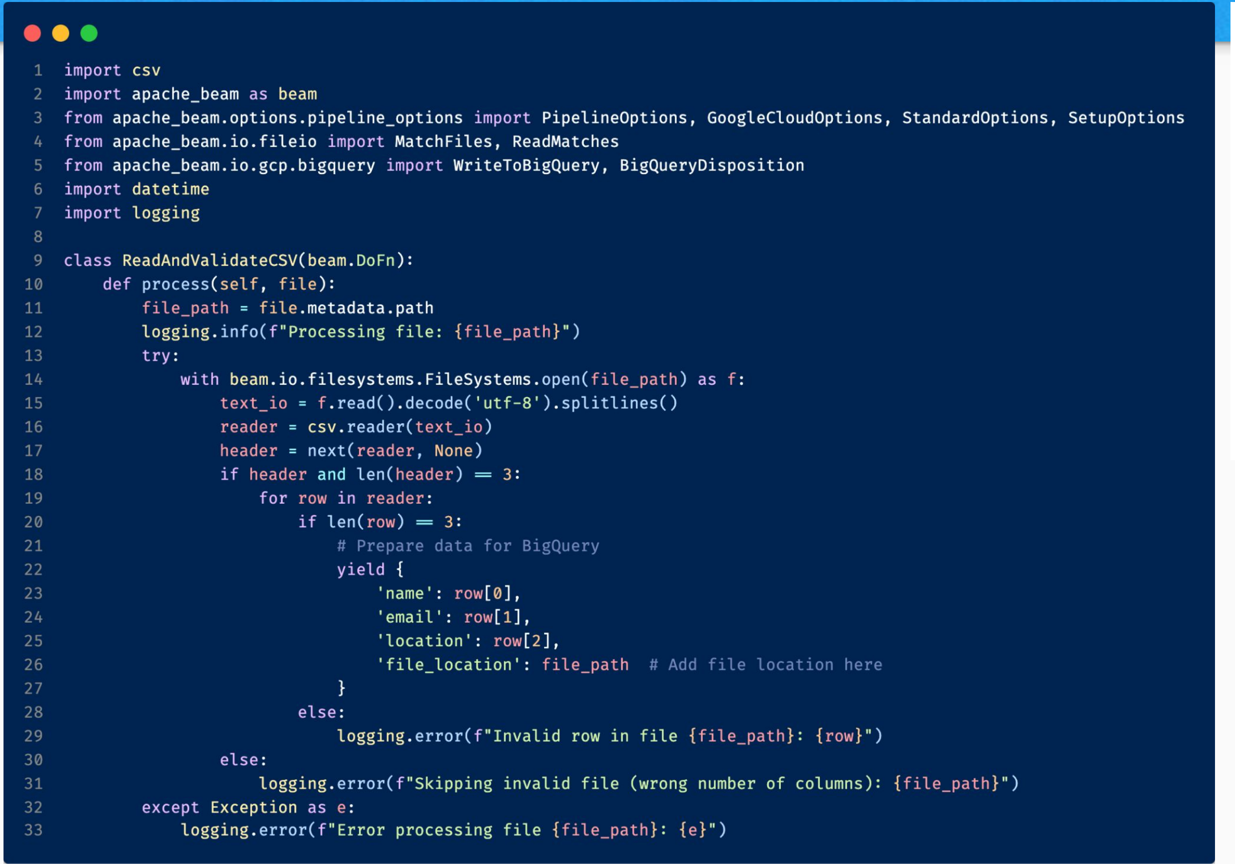
+ Ease of use
  + Integration with other GCP services
  + One-click deployment
  + Create/Update/Delete
  + UI/gcloud/API/Terraform
+ Infrastructure scalability (GKE, GAE, CloudSQL)
+ Logging and Monitoring
+ Security
  + Webserver: behind IAP, Network ACLs
  + Shared VPC, Private IP
  + VPC SC (Beta)

+ Operability & Maintainability
  + Python dependency management
  + Airflow config update propagation
  + In-place version upgrades (Beta)

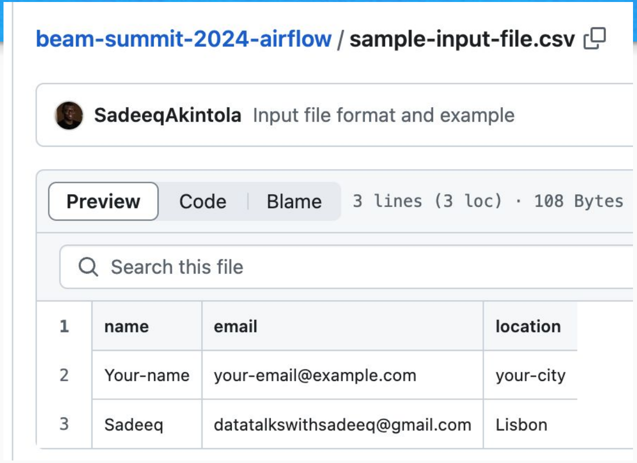
- Some Airflow capabilities are not available

Code Review (Beam Pipeline)

Importing Libraries and Reading Files



```python
import csv
import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions, GoogleCloudOptions, StandardOptions, SetupOptions
from apache_beam.io.fileio import MatchFiles, ReadMatches
from apache_beam.io.gcp.bigquery import WriteToBigQuery, BigQueryDisposition
import datetime
import logging

class ReadAndValidateCSV(beam.DoFn):
    def process(self, file):
        file_path = file.metadata.path
        logging.info(f"Processing file: {file_path}")
        try:
            with beam.io.filesystems.FileSystems.open(file_path) as f:
                text_io = f.read().decode('utf-8').splitlines()
                reader = csv.reader(text_io)
                header = next(reader, None)
                if header and len(header) == 3:
                    for row in reader:
                        if len(row) == 3:
                            # Prepare data for BigQuery
                            yield {
                                'name': row[0],
                                'email': row[1],
                                'location': row[2],
                                'file_location': file_path  # Add file location here
                            }
                        else:
                            logging.error(f"Invalid row in file {file_path}: {row}")
                else:
                    logging.error(f"Skipping invalid file (wrong number of columns): {file_path}")
        except Exception as e:
            logging.error(f"Error processing file {file_path}: {e}")
```

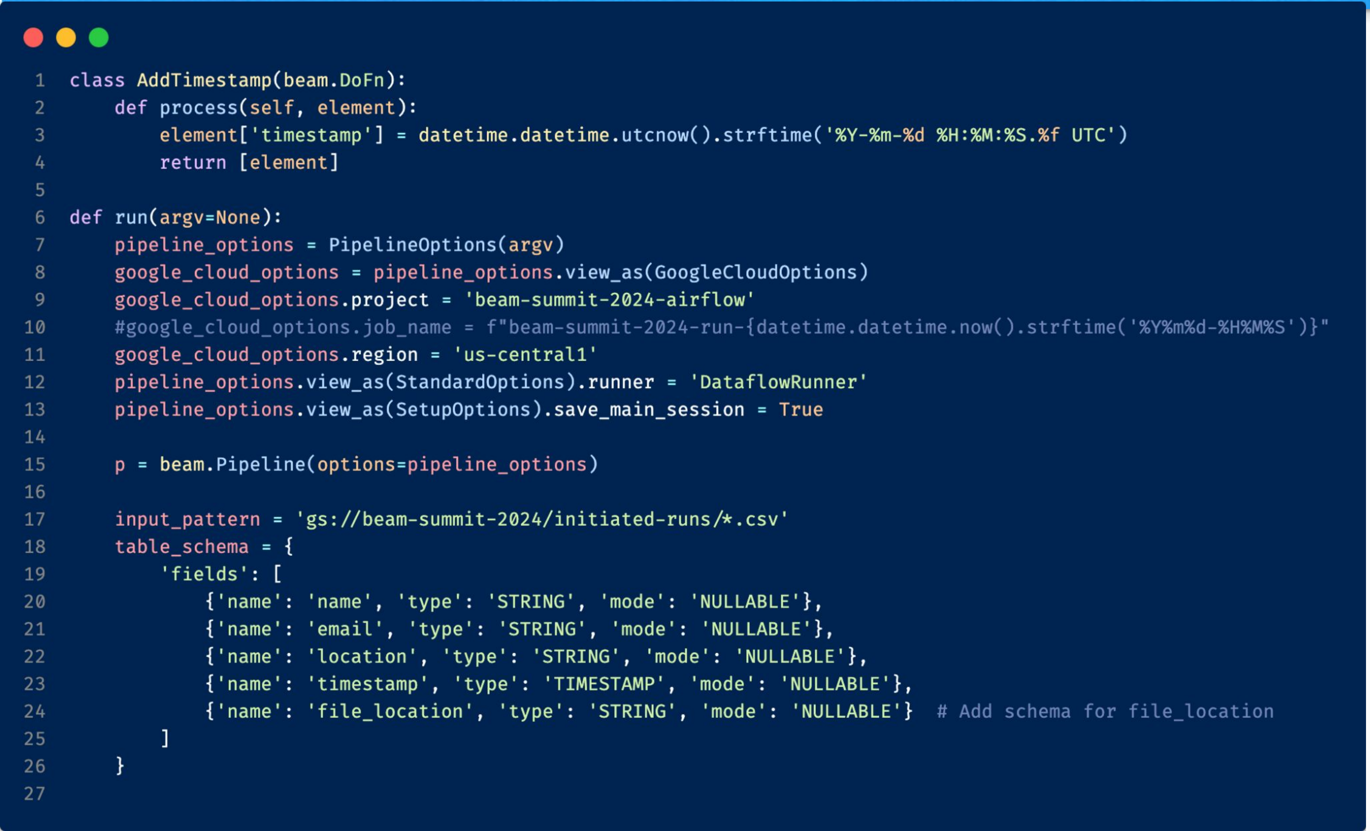


beam-summit-2024-airflow / sample-input-file.csv

SadeeqAkintola Input file format and example

1	name	email	location
2	Your-name	your-email@example.com	your-city
3	Sadeeq	datatalkswithsadeeq@gmail.com	Lisbon

Building the Functions for the Pipeline



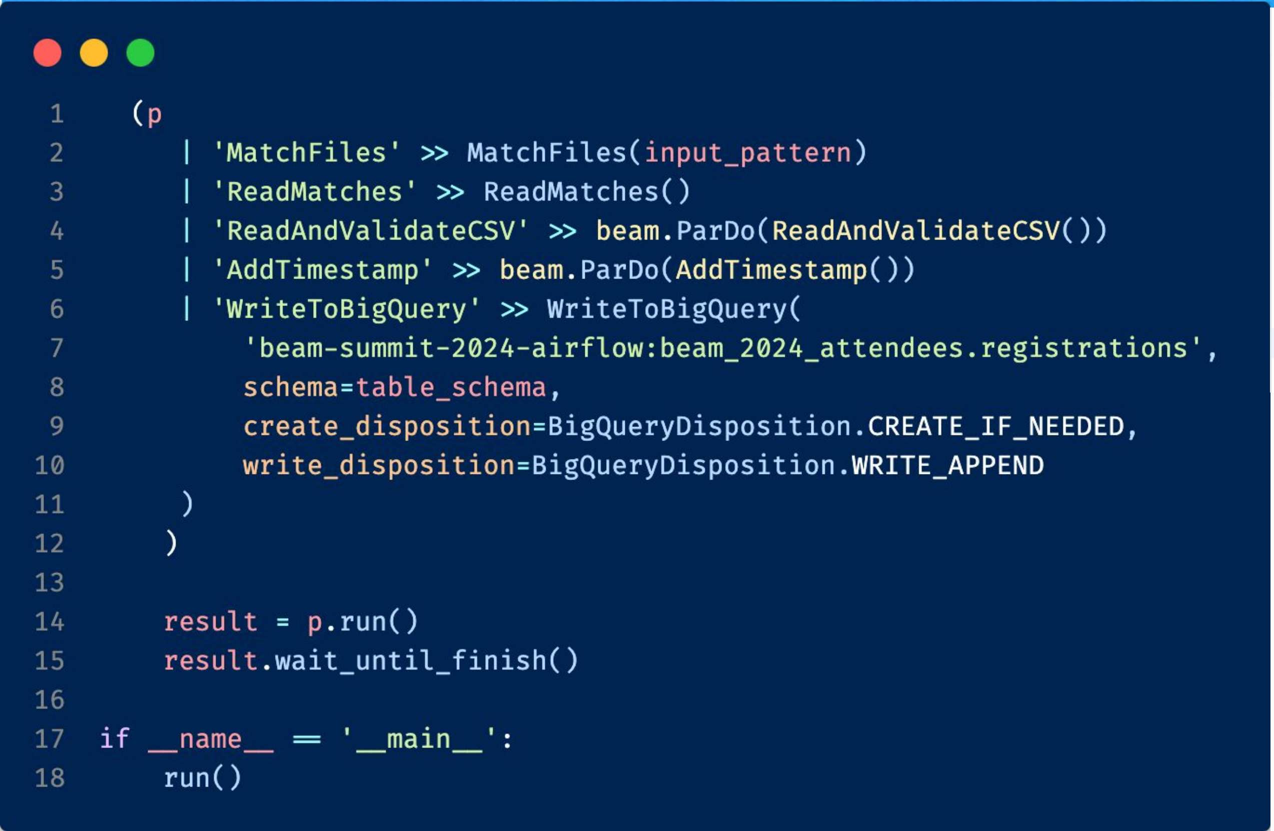
```python
class AddTimestamp(beam.DoFn):
    def process(self, element):
        element['timestamp'] = datetime.datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S.%f UTC')
        return [element]

def run(argv=None):
    pipeline_options = PipelineOptions(argv)
    google_cloud_options = pipeline_options.view_as(GoogleCloudOptions)
    google_cloud_options.project = 'beam-summit-2024-airflow'
    #google_cloud_options.job_name = f"beam-summit-2024-run-{datetime.datetime.now().strftime('%Y%m%d-%H%M%S')}"
    google_cloud_options.region = 'us-central1'
    pipeline_options.view_as(StandardOptions).runner = 'DataflowRunner'
    pipeline_options.view_as(SetupOptions).save_main_session = True

    p = beam.Pipeline(options=pipeline_options)

    input_pattern = 'gs://beam-summit-2024/initiated-runs/*.csv'
    table_schema = {
        'fields': [
            {'name': 'name', 'type': 'STRING', 'mode': 'NULLABLE'},
            {'name': 'email', 'type': 'STRING', 'mode': 'NULLABLE'},
            {'name': 'location', 'type': 'STRING', 'mode': 'NULLABLE'},
            {'name': 'timestamp', 'type': 'TIMESTAMP', 'mode': 'NULLABLE'},
            {'name': 'file_location', 'type': 'STRING', 'mode': 'NULLABLE'}  # Add schema for file_location
        ]
    }

```

Worth Noting

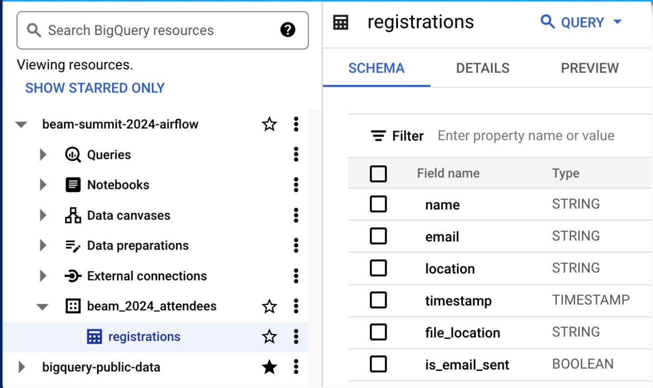
- **beam.DoFn :** The DoFn object that you pass to ParDo contains the processing logic that gets applied to the elements in the input collection.
- **PipelineOptions :** are used to configure Pipelines. You can extend PipelineOptions to create custom configuration options.
- **beam.Pipeline :** is the entry point for constructing and running a data processing pipeline, defining the series of transformations and operations that will be executed on the input data.

BEAM SUMMIT

Bringing it all together, and Run!



```
  (p
     | 'MatchFiles' >> MatchFiles(input_pattern)
     | 'ReadMatches' >> ReadMatches()
     | 'ReadAndValidateCSV' >> beam.ParDo(ReadAndValidateCSV())
     | 'AddTimestamp' >> beam.ParDo(AddTimestamp())
     | 'WriteToBigQuery' >> WriteToBigQuery(
         'beam-summit-2024-airflow:beam_2024_attendees.registrations',
         schema=table_schema,
         create_disposition=BigQueryDisposition.CREATE_IF_NEEDED,
         write_disposition=BigQueryDisposition.WRITE_APPEND
     )
    )

    result = p.run()
    result.wait_until_finish()

if __name__ == '__main__':
    run()
```



Search BigQuery resources

Viewing resources.

SHOW STARRED ONLY

- beam-summit-2024-airflow
  - Queries
  - Notebooks
  - Data canvases
  - Data preparations
  - External connections
  - beam_2024_attendees
    - registrations
- bigquery-public-data

registrations — QUERY

SCHEMA | DETAILS | PREVIEW

Filter: Enter property name or value

Field name	Type
name	STRING
email	STRING
location	STRING
timestamp	TIMESTAMP
file_location	STRING
is_email_sent	BOOLEAN

Worth Noting

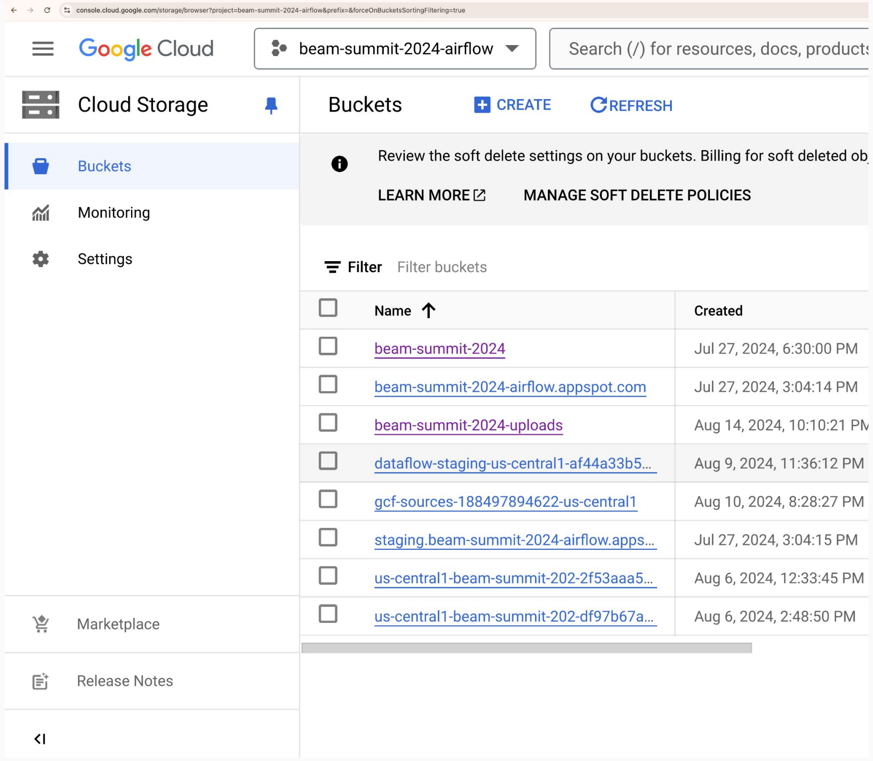
- **The '|' symbol** is used as an operator to apply transformations to a PCollection

- **WriteToBigQuery :** is a transform used to write data from a PCollection to a BigQuery table.

BEAM SUMMIT

Results (Beam) - GCS Folders

Before the Pipeline Run

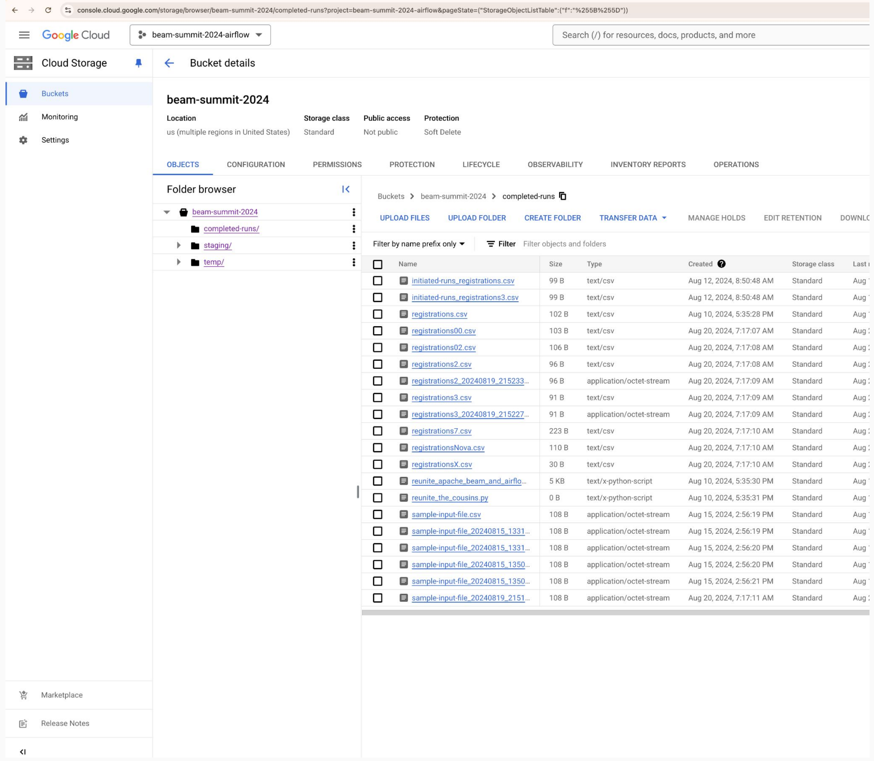


console.cloud.google.com/storage/browser?project=beam-summit-2024-airflow&prefix=&forceOnBucketsSortingFiltering=true

Review the soft delete settings on your buckets. Billing for soft deleted ob

Name	Created
beam-summit-2024	Jul 27, 2024, 6:30:00 PM
beam-summit-2024-airflow.appspot.com	Jul 27, 2024, 3:04:14 PM
beam-summit-2024-uploads	Aug 14, 2024, 10:10:21 PM
dataflow-staging-us-central1-af44a33b5…	Aug 9, 2024, 11:36:12 PM
gcf-sources-188497894622-us-central1	Aug 10, 2024, 8:28:27 PM
staging.beam-summit-2024-airflow.apps…	Jul 27, 2024, 3:04:15 PM
us-central1-beam-summit-202-2f53aaa5…	Aug 6, 2024, 12:33:45 PM
us-central1-beam-summit-202-df97b67a…	Aug 6, 2024, 2:48:50 PM

After the Pipeline Run



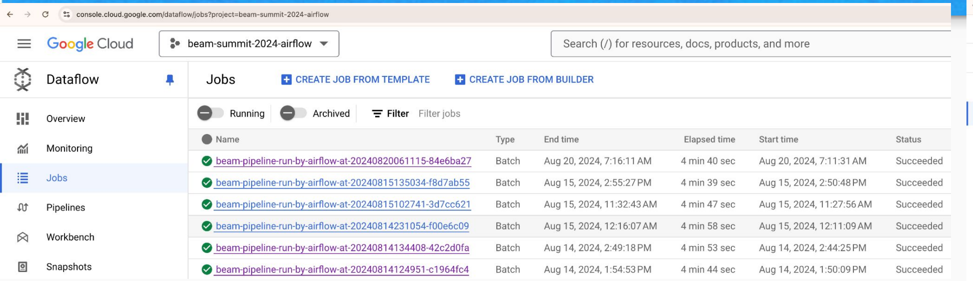
console.cloud.google.com/storage/browser/beam-summit-2024/completed-runs?project=beam-summit-2024-airflow&pageState=("StorageObjectListTable":("f":"%255B%255D"))

beam-summit-2024

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	Soft Delete

Name	Size	Type	Created	Storage class	Last
initiated-runs_registrations.csv	99 B	text/csv	Aug 12, 2024, 8:50:48 AM	Standard	Aug
initiated-runs_registrations3.csv	99 B	text/csv	Aug 12, 2024, 8:50:48 AM	Standard	Aug
registrations.csv	102 B	text/csv	Aug 10, 2024, 5:35:28 PM	Standard	Aug
registrations00.csv	103 B	text/csv	Aug 20, 2024, 7:17:07 AM	Standard	Aug
registrations02.csv	106 B	text/csv	Aug 20, 2024, 7:17:08 AM	Standard	Aug
registrations2.csv	96 B	text/csv	Aug 20, 2024, 7:17:08 AM	Standard	Aug
registrations2_20240819_215233…	96 B	application/octet-stream	Aug 20, 2024, 7:17:09 AM	Standard	Aug
registrations3.csv	91 B	text/csv	Aug 20, 2024, 7:17:09 AM	Standard	Aug
registrations3_20240819_215227…	91 B	application/octet-stream	Aug 20, 2024, 7:17:09 AM	Standard	Aug
registrations7.csv	223 B	text/csv	Aug 20, 2024, 7:17:10 AM	Standard	Aug
registrationsNova.csv	110 B	text/csv	Aug 20, 2024, 7:17:10 AM	Standard	Aug
registrationsX.csv	30 B	text/csv	Aug 20, 2024, 7:17:10 AM	Standard	Aug
reunite_apache_beam_and_airflo…	5 KB	text/x-python-script	Aug 10, 2024, 5:35:30 PM	Standard	Aug
reunite_the_cousins.py	0 B	text/x-python-script	Aug 10, 2024, 5:35:31 PM	Standard	Aug
sample-input-file.csv	108 B	application/octet-stream	Aug 15, 2024, 2:56:19 PM	Standard	Aug
sample-input-file_20240815_1331…	108 B	application/octet-stream	Aug 15, 2024, 2:56:19 PM	Standard	Aug
sample-input-file_20240815_1331…	108 B	application/octet-stream	Aug 15, 2024, 2:56:20 PM	Standard	Aug
sample-input-file_20240815_1350…	108 B	application/octet-stream	Aug 15, 2024, 2:56:20 PM	Standard	Aug
sample-input-file_20240815_1350…	108 B	application/octet-stream	Aug 15, 2024, 2:56:21 PM	Standard	Aug
sample-input-file_20240819_2151…	108 B	application/octet-stream	Aug 20, 2024, 7:17:11 AM	Standard	Aug

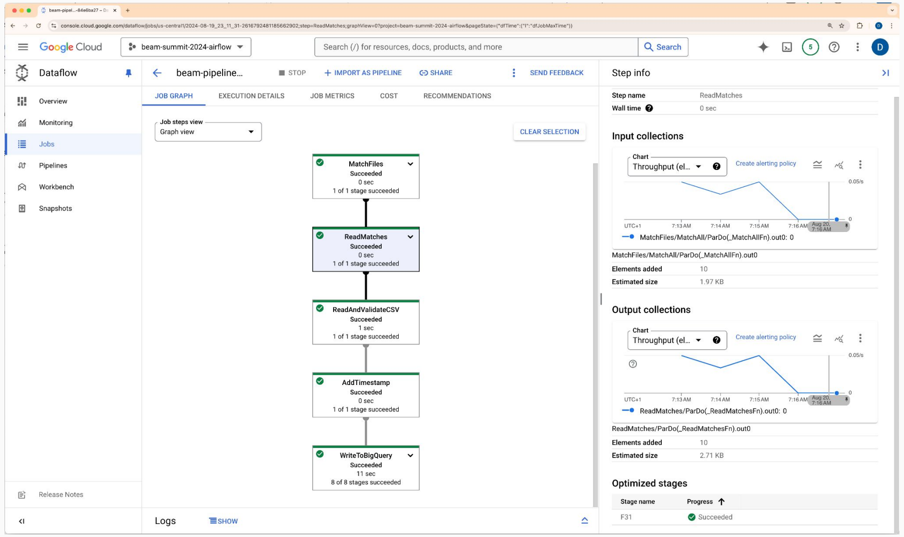
Results (Beam) - Dataflow and BigQuery



console.cloud.google.com/dataflow/jobs?project=beam-summit-2024-airflow

Dataflow — Jobs

Name	Type	End time	Elapsed time	Start time	Status
beam-pipeline-run-by-airflow-at-20240820061115-84e6ba27	Batch	Aug 20, 2024, 7:16:11 AM	4 min 40 sec	Aug 20, 2024, 7:11:31 AM	Succeeded
beam-pipeline-run-by-airflow-at-20240815135034-f8d7ab55	Batch	Aug 15, 2024, 2:55:27 PM	4 min 39 sec	Aug 15, 2024, 2:50:48 PM	Succeeded
beam-pipeline-run-by-airflow-at-20240815102741-3d7cc621	Batch	Aug 15, 2024, 11:32:43 AM	4 min 47 sec	Aug 15, 2024, 11:27:56 AM	Succeeded
beam-pipeline-run-by-airflow-at-20240814231054-f00e6c09	Batch	Aug 15, 2024, 12:16:07 AM	4 min 58 sec	Aug 15, 2024, 12:11:09 AM	Succeeded
beam-pipeline-run-by-airflow-at-20240814134408-42c2d0fa	Batch	Aug 14, 2024, 2:49:18 PM	4 min 53 sec	Aug 14, 2024, 2:44:25 PM	Succeeded
beam-pipeline-run-by-airflow-at-20240814124951-c1964fc4	Batch	Aug 14, 2024, 1:54:53 PM	4 min 44 sec	Aug 14, 2024, 1:50:09 PM	Succeeded



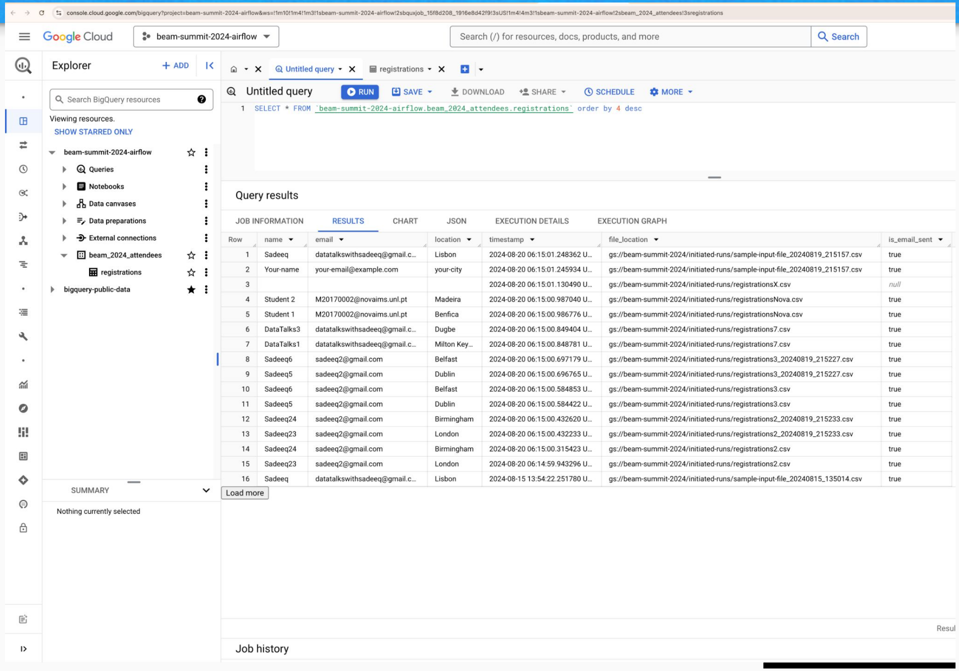
Step info

Step name: ReadMatches
Wall time: 0 sec

Input collections: MatchFiles/MatchAll/ParDo(_MatchAllFn).out0
Elements added: 10
Estimated size: 1.97 KB

Output collections: ReadMatches/ParDo(_ReadMatchesFn).out0
Elements added: 10
Estimated size: 2.71 KB

Optimized stages: F31 — Succeeded



```
SELECT * FROM `beam-summit-2024-airflow.beam_2024_attendees.registrations` order by 4 desc
```

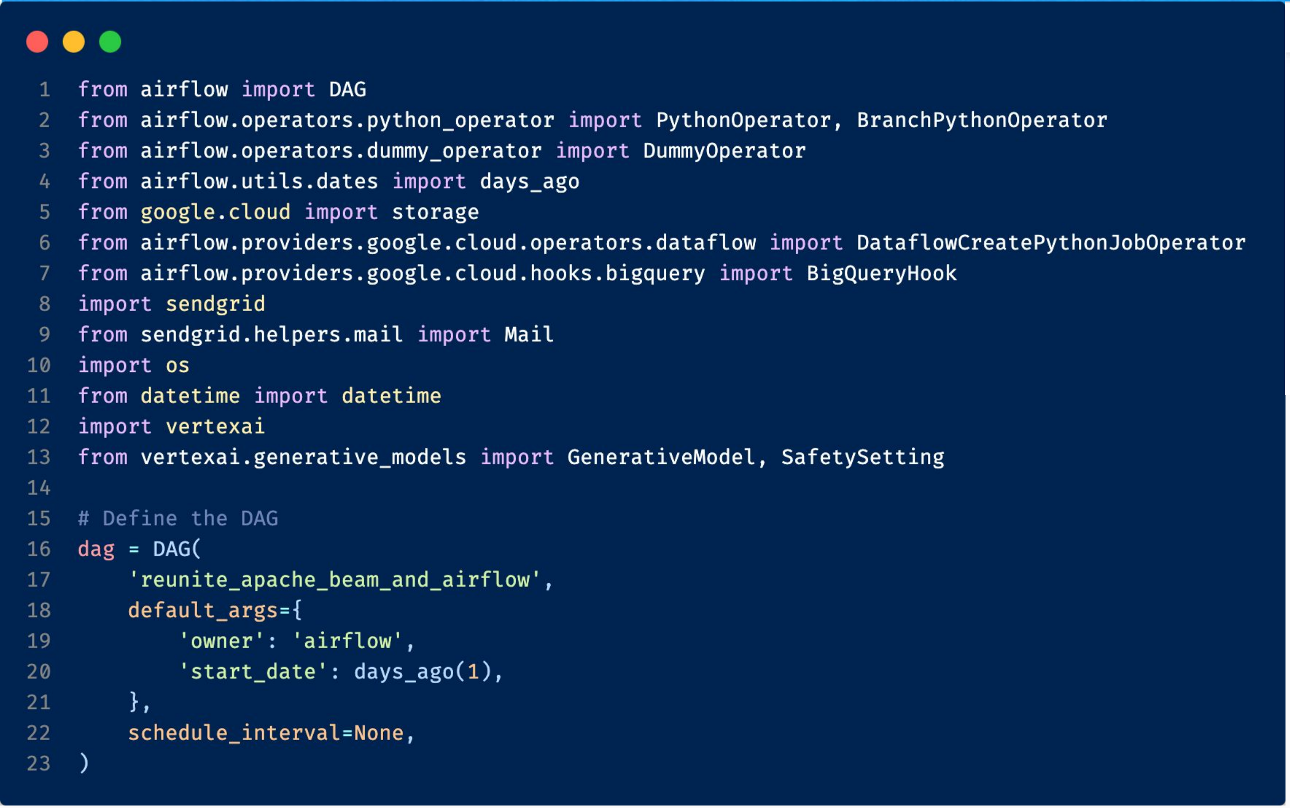
Query results

Row	name	email	location	timestamp	file_location	is_email_sent
1	Sadeeq	datatalkswithsadeeq@gmail.c...	Lisbon	2024-08-20 06:15:01.248362 U...	gs://beam-summit-2024/initiated-runs/sample-input-file_20240819_215157.csv	true
2	Your-name	your-email@example.com	your-city	2024-08-20 06:15:01.245934 U...	gs://beam-summit-2024/initiated-runs/sample-input-file_20240819_215157.csv	true
3				2024-08-20 06:15:01.130490 U...	gs://beam-summit-2024/initiated-runs/registrationsX.csv	null
4	Student 2	M20170002@novaims.unl.pt	Madeira	2024-08-20 06:15:00.987040 U...	gs://beam-summit-2024/initiated-runs/registrationsNova.csv	true
5	Student 1	M20170002@novaims.unl.pt	Benfica	2024-08-20 06:15:00.986776 U...	gs://beam-summit-2024/initiated-runs/registrationsNova.csv	true
6	DataTalks3	datatalkswithsadeeq@gmail.c...	Dugbe	2024-08-20 06:15:00.849404 U...	gs://beam-summit-2024/initiated-runs/registrations7.csv	true
7	DataTalks1	datatalkswithsadeeq@gmail.c...	Milton Key...	2024-08-20 06:15:00.848781 U...	gs://beam-summit-2024/initiated-runs/registrations7.csv	true
8	Sadeeq6	sadeeq2@gmail.com	Belfast	2024-08-20 06:15:00.697179 U...	gs://beam-summit-2024/initiated-runs/registrations3_20240819_215227.csv	true
9	Sadeeq5	sadeeq2@gmail.com	Dublin	2024-08-20 06:15:00.696765 U...	gs://beam-summit-2024/initiated-runs/registrations3_20240819_215227.csv	true
10	Sadeeq6	sadeeq2@gmail.com	Belfast	2024-08-20 06:15:00.584853 U...	gs://beam-summit-2024/initiated-runs/registrations3.csv	true
11	Sadeeq5	sadeeq2@gmail.com	Dublin	2024-08-20 06:15:00.584422 U...	gs://beam-summit-2024/initiated-runs/registrations3.csv	true
12	Sadeeq24	sadeeq2@gmail.com	Birmingham	2024-08-20 06:15:00.432620 U...	gs://beam-summit-2024/initiated-runs/registrations2_20240819_215233.csv	true
13	Sadeeq23	sadeeq2@gmail.com	London	2024-08-20 06:15:00.432233 U...	gs://beam-summit-2024/initiated-runs/registrations2_20240819_215233.csv	true
14	Sadeeq24	sadeeq2@gmail.com	Birmingham	2024-08-20 06:15:00.315423 U...	gs://beam-summit-2024/initiated-runs/registrations2.csv	true
15	Sadeeq23	sadeeq2@gmail.com	London	2024-08-20 06:14:59.943296 U...	gs://beam-summit-2024/initiated-runs/registrations2.csv	true
16	Sadeeq	datatalkswithsadeeq@gmail.c...	Lisbon	2024-08-15 13:54:22.251780 U...	gs://beam-summit-2024/initiated-runs/sample-input-file_20240815_135014.csv	true

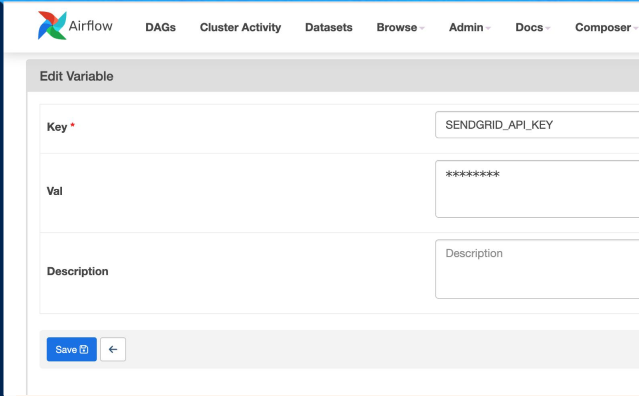
BEAM SUMMIT

Code Review (Airflow DAG)

Import Libraries, Create the DAG, and Input the Environment Variables in Airflow UI

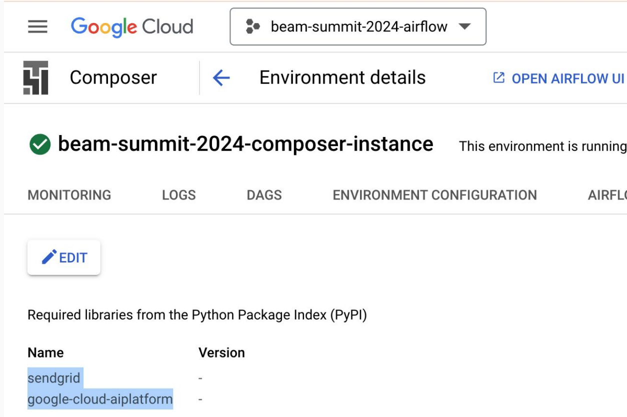


```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator, BranchPythonOperator
from airflow.operators.dummy_operator import DummyOperator
from airflow.utils.dates import days_ago
from google.cloud import storage
from airflow.providers.google.cloud.operators.dataflow import DataflowCreatePythonJobOperator
from airflow.providers.google.cloud.hooks.bigquery import BigQueryHook
import sendgrid
from sendgrid.helpers.mail import Mail
import os
from datetime import datetime
import vertexai
from vertexai.generative_models import GenerativeModel, SafetySetting

# Define the DAG
dag = DAG(
    'reunite_apache_beam_and_airflow',
    default_args={
        'owner': 'airflow',
        'start_date': days_ago(1),
    },
    schedule_interval=None,
)
```



Edit Variable

Key *	SENDGRID_API_KEY
Val	********
Description	Description



beam-summit-2024-composer-instance This environment is running

Required libraries from the Python Package Index (PyPI)

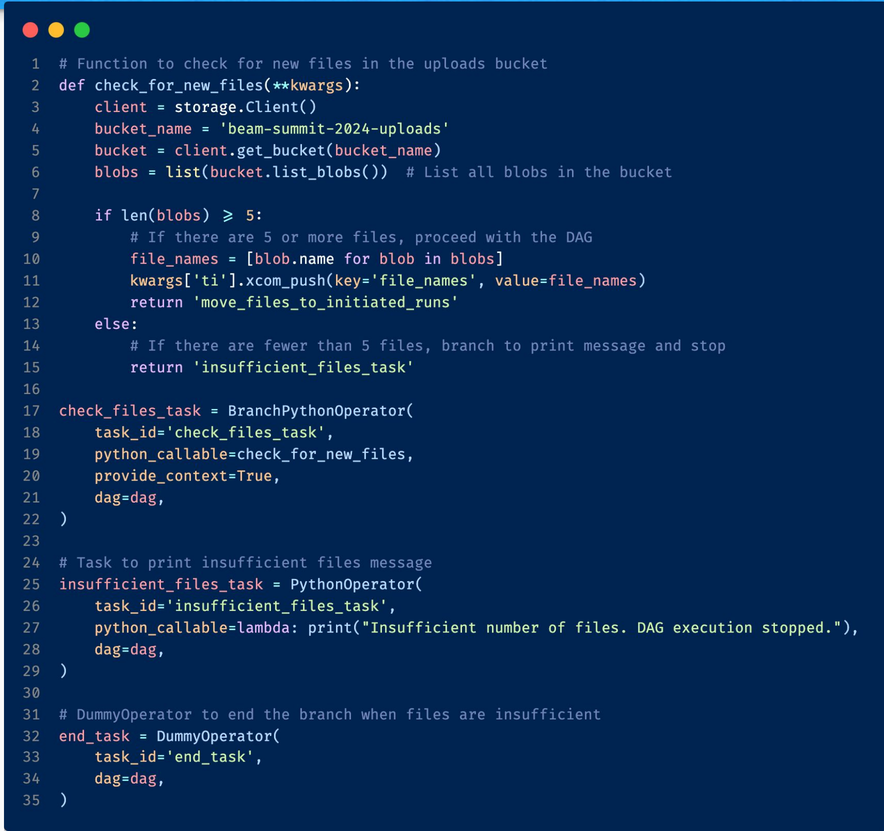
Name	Version
sendgrid	-
google-cloud-aiplatform	-

Operators! Operators!! Operators!!!

- **The PythonOperator**: executes Python functions as tasks in a DAG, allowing for flexible workflow management and integration with other tasks. It supports arguments, retries, and logging.

- **The BranchPythonOperator:** allows you to conditionally direct the execution flow of a DAG. It runs a Python function that returns the ***taskID of the next task to execute***, effectively branching the workflow. Only the branch selected runs, and downstream tasks are determined by this choice, allowing for ***dynamic workflows*** based on runtime conditions.

- **The DummyOperator:** is a no-op operator used primarily as a placeholder in DAGs. ***It doesn't perform any action*** but can be useful for organizing complex workflows, acting as a boundary, or grouping tasks without executing any tasks itself. It's often used for joining or splitting task flows or as a marker in a DAG's structure.

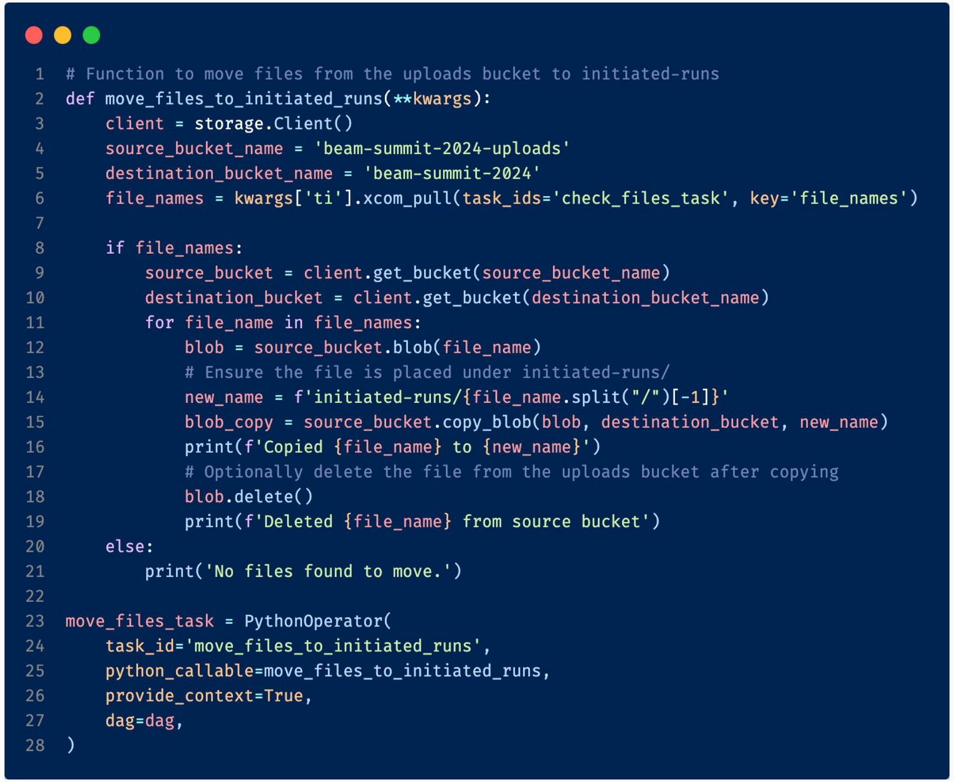


```python
# Function to check for new files in the uploads bucket
def check_for_new_files(**kwargs):
    client = storage.Client()
    bucket_name = 'beam-summit-2024-uploads'
    bucket = client.get_bucket(bucket_name)
    blobs = list(bucket.list_blobs())  # List all blobs in the bucket

    if len(blobs) ≥ 5:
        # If there are 5 or more files, proceed with the DAG
        file_names = [blob.name for blob in blobs]
        kwargs['ti'].xcom_push(key='file_names', value=file_names)
        return 'move_files_to_initiated_runs'
    else:
        # If there are fewer than 5 files, branch to print message and stop
        return 'insufficient_files_task'

check_files_task = BranchPythonOperator(
    task_id='check_files_task',
    python_callable=check_for_new_files,
    provide_context=True,
    dag=dag,
)

# Task to print insufficient files message
insufficient_files_task = PythonOperator(
    task_id='insufficient_files_task',
    python_callable=lambda: print("Insufficient number of files. DAG execution stopped."),
    dag=dag,
)

# DummyOperator to end the branch when files are insufficient
end_task = DummyOperator(
    task_id='end_task',
    dag=dag,
)
```

Explaining the *xcom* Libraries

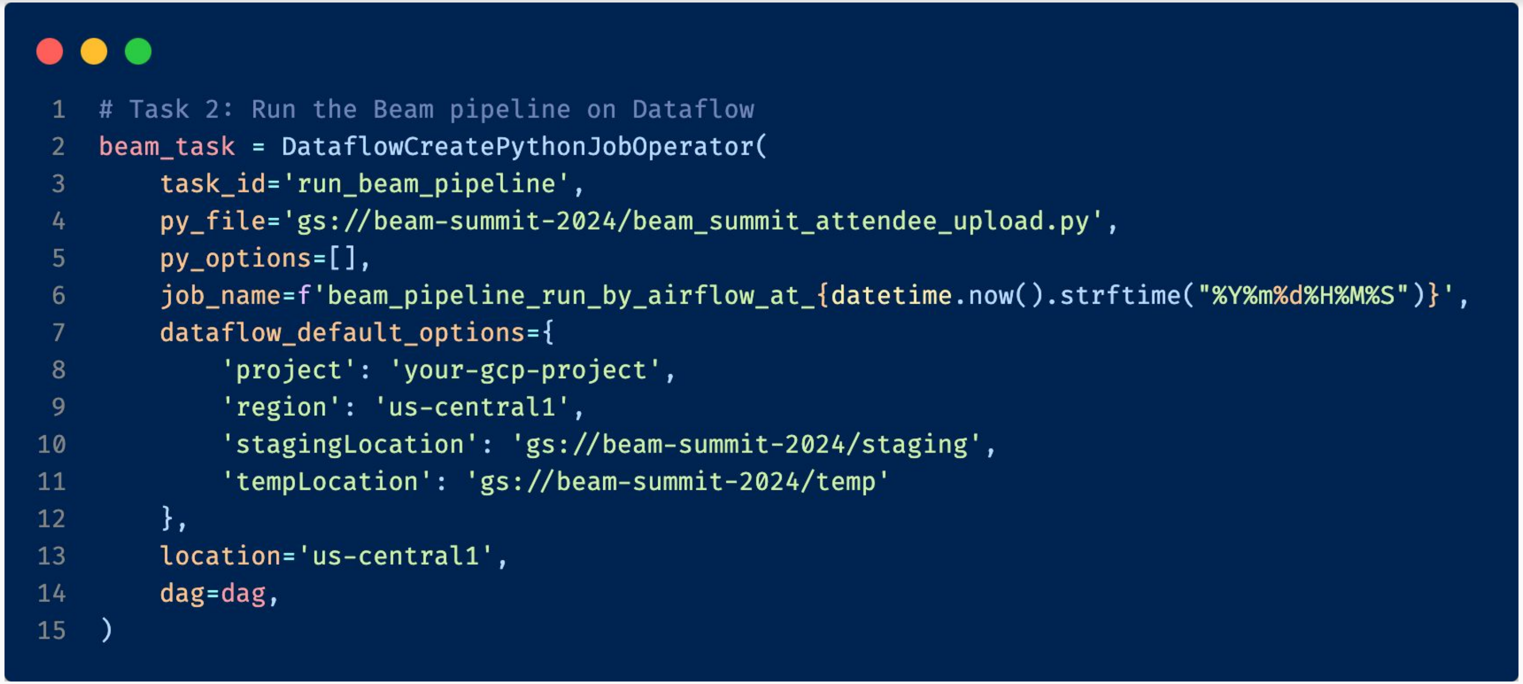
The ***xcom_pull*** is a method in Apache Airflow that allows a task to retrieve data (XCom) pushed by a previous task. It can pull specific data based on task ID, key, and execution date. It's typically used for inter-task communication within a DAG to share information across tasks.

- **Purpose**: xcom_pull retrieves shared data (XComs) between tasks in a DAG.
- **Communication**: Allows tasks to share data using xcom_push and xcom_pull.
- **Task ID**: Pull data from a specific task by specifying its task_id.
- **Key-Value:** Retrieve specific data by providing a key; defaults to all if not specified.
- **Execution Date:** Access data from a specific execution date if needed.
- **Return Value:** Returns the relevant data or a list if multiple records match.
- **Usage:** Commonly used in task functions/operators for dynamic workflows.
- **Limitations:** Best for small data; use external storage for large data transfers.



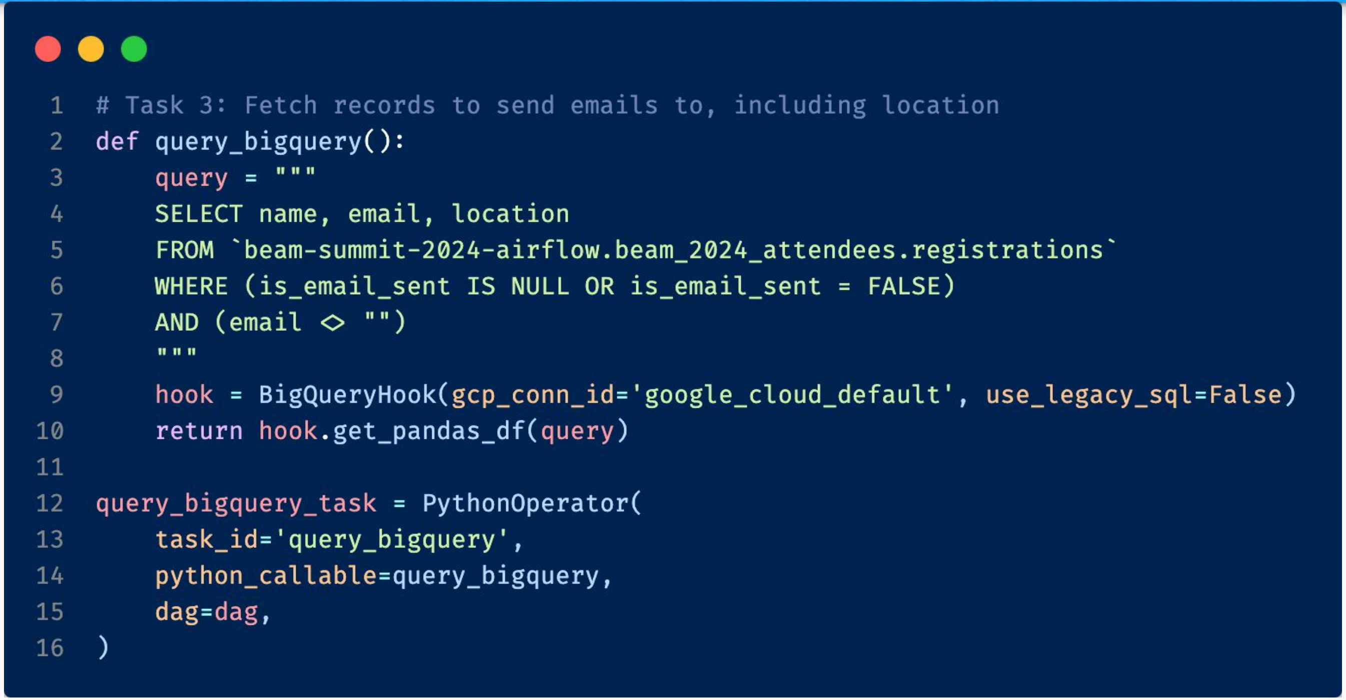
```python
# Function to move files from the uploads bucket to initiated-runs
def move_files_to_initiated_runs(**kwargs):
    client = storage.Client()
    source_bucket_name = 'beam-summit-2024-uploads'
    destination_bucket_name = 'beam-summit-2024'
    file_names = kwargs['ti'].xcom_pull(task_ids='check_files_task', key='file_names')

    if file_names:
        source_bucket = client.get_bucket(source_bucket_name)
        destination_bucket = client.get_bucket(destination_bucket_name)
        for file_name in file_names:
            blob = source_bucket.blob(file_name)
            # Ensure the file is placed under initiated-runs/
            new_name = f'initiated-runs/{file_name.split("/")[-1]}'
            blob_copy = source_bucket.copy_blob(blob, destination_bucket, new_name)
            print(f'Copied {file_name} to {new_name}')
            # Optionally delete the file from the uploads bucket after copying
            blob.delete()
            print(f'Deleted {file_name} from source bucket')
    else:
        print('No files found to move.')

move_files_task = PythonOperator(
    task_id='move_files_to_initiated_runs',
    python_callable=move_files_to_initiated_runs,
    provide_context=True,
    dag=dag,
)
```

This code block is the reason for this talk!



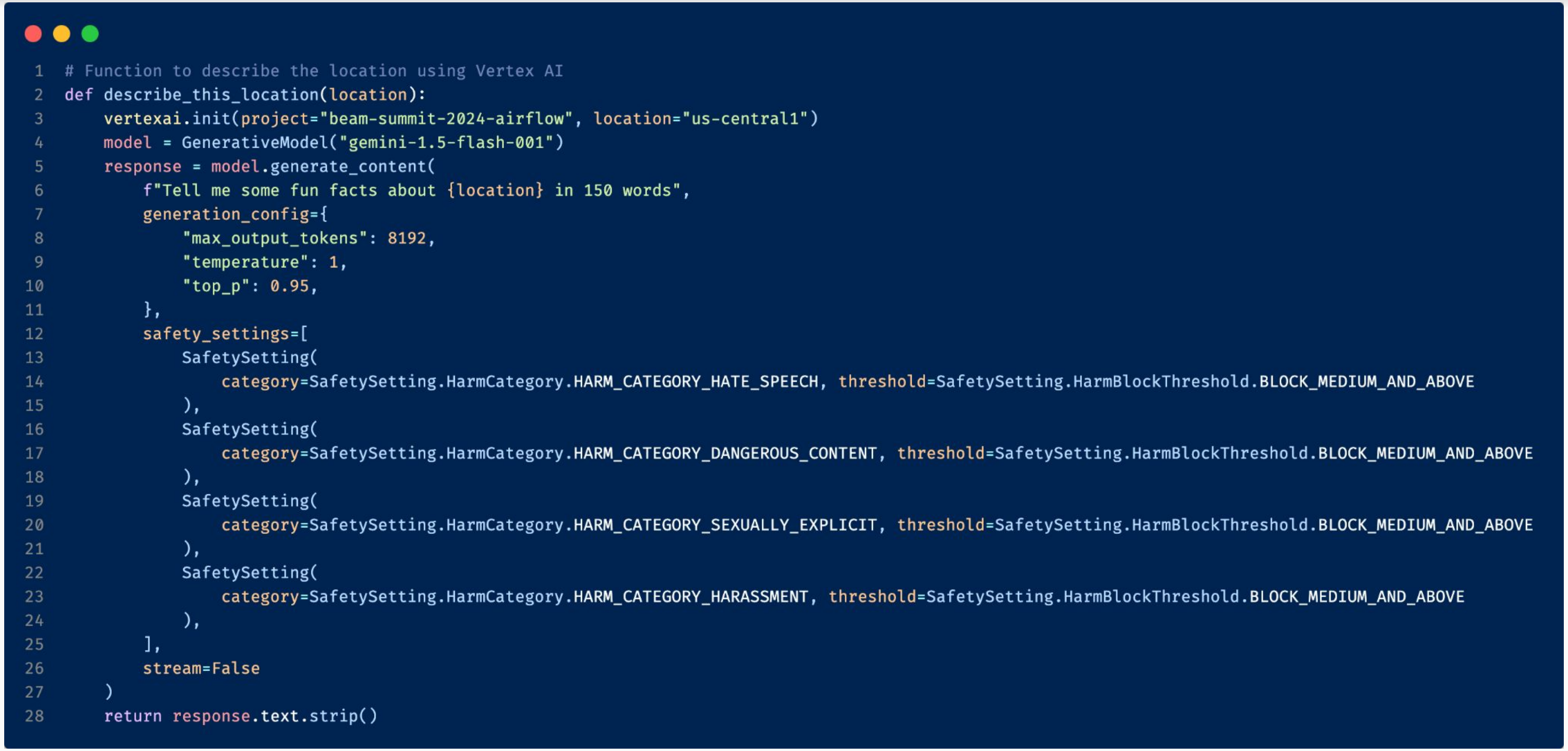
```python
# Task 2: Run the Beam pipeline on Dataflow
beam_task = DataflowCreatePythonJobOperator(
    task_id='run_beam_pipeline',
    py_file='gs://beam-summit-2024/beam_summit_attendee_upload.py',
    py_options=[],
    job_name=f'beam_pipeline_run_by_airflow_at_{datetime.now().strftime("%Y%m%d%H%M%S")}',
    dataflow_default_options={
        'project': 'your-gcp-project',
        'region': 'us-central1',
        'stagingLocation': 'gs://beam-summit-2024/staging',
        'tempLocation': 'gs://beam-summit-2024/temp'
    },
    location='us-central1',
    dag=dag,
)
```

Fetching data from BigQuery in Airflow

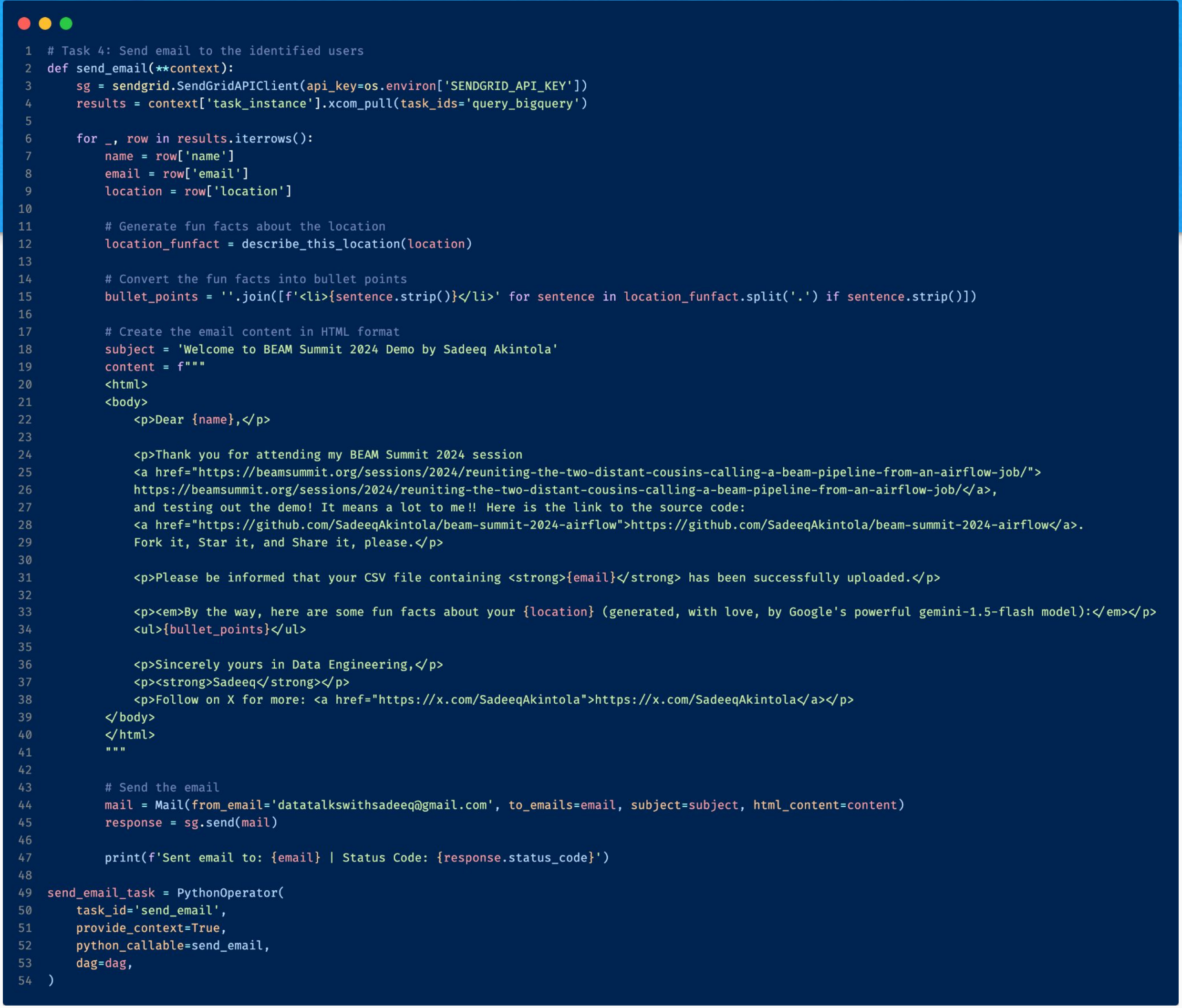


```python
# Task 3: Fetch records to send emails to, including location
def query_bigquery():
    query = """
    SELECT name, email, location
    FROM `beam-summit-2024-airflow.beam_2024_attendees.registrations`
    WHERE (is_email_sent IS NULL OR is_email_sent = FALSE)
    AND (email <> "")
    """
    hook = BigQueryHook(gcp_conn_id='google_cloud_default', use_legacy_sql=False)
    return hook.get_pandas_df(query)

query_bigquery_task = PythonOperator(
    task_id='query_bigquery',
    python_callable=query_bigquery,
    dag=dag,
)
```

Creating a Gemini Flash GenAI Model in Airflow



```python
# Function to describe the location using Vertex AI
def describe_this_location(location):
    vertexai.init(project="beam-summit-2024-airflow", location="us-central1")
    model = GenerativeModel("gemini-1.5-flash-001")
    response = model.generate_content(
        f"Tell me some fun facts about {location} in 150 words",
        generation_config={
            "max_output_tokens": 8192,
            "temperature": 1,
            "top_p": 0.95,
        },
        safety_settings=[
            SafetySetting(
                category=SafetySetting.HarmCategory.HARM_CATEGORY_HATE_SPEECH, threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
            ),
            SafetySetting(
                category=SafetySetting.HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT, threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
            ),
            SafetySetting(
                category=SafetySetting.HarmCategory.HARM_CATEGORY_SEXUALLY_EXPLICIT, threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
            ),
            SafetySetting(
                category=SafetySetting.HarmCategory.HARM_CATEGORY_HARASSMENT, threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
            ),
        ],
        stream=False
    )
    return response.text.strip()
```



```python
# Task 4: Send email to the identified users
def send_email(**context):
    sg = sendgrid.SendGridAPIClient(api_key=os.environ['SENDGRID_API_KEY'])
    results = context['task_instance'].xcom_pull(task_ids='query_bigquery')

    for _, row in results.iterrows():
        name = row['name']
        email = row['email']
        location = row['location']

        # Generate fun facts about the location
        location_funfact = describe_this_location(location)

        # Convert the fun facts into bullet points
        bullet_points = ''.join([f'<li>{sentence.strip()}</li>' for sentence in location_funfact.split('.') if sentence.strip()])

        # Create the email content in HTML format
        subject = 'Welcome to BEAM Summit 2024 Demo by Sadeeq Akintola'
        content = f"""
        <html>
        <body>
            <p>Dear {name},</p>

            <p>Thank you for attending my BEAM Summit 2024 session
            <a href="https://beamsummit.org/sessions/2024/reuniting-the-two-distant-cousins-calling-a-beam-pipeline-from-an-airflow-job/">
            https://beamsummit.org/sessions/2024/reuniting-the-two-distant-cousins-calling-a-beam-pipeline-from-an-airflow-job/</a>,
            and testing out the demo! It means a lot to me!! Here is the link to the source code:
            <a href="https://github.com/SadeeqAkintola/beam-summit-2024-airflow">https://github.com/SadeeqAkintola/beam-summit-2024-airflow</a>.
            Fork it, Star it, and Share it, please.</p>

            <p>Please be informed that your CSV file containing <strong>{email}</strong> has been successfully uploaded.</p>

            <p><em>By the way, here are some fun facts about your {location} (generated, with love, by Google's powerful gemini-1.5-flash model):</em></p>
            <ul>{bullet_points}</ul>

            <p>Sincerely yours in Data Engineering,</p>
            <p><strong>Sadeeq</strong></p>
            <p>Follow on X for more: <a href="https://x.com/SadeeqAkintola">https://x.com/SadeeqAkintola</a></p>
        </body>
        </html>
        """

        # Send the email
        mail = Mail(from_email='datatalkswithsadeeq@gmail.com', to_emails=email, subject=subject, html_content=content)
        response = sg.send(mail)

        print(f'Sent email to: {email} | Status Code: {response.status_code}')

send_email_task = PythonOperator(
    task_id='send_email',
    provide_context=True,
    python_callable=send_email,
    dag=dag,
)
```

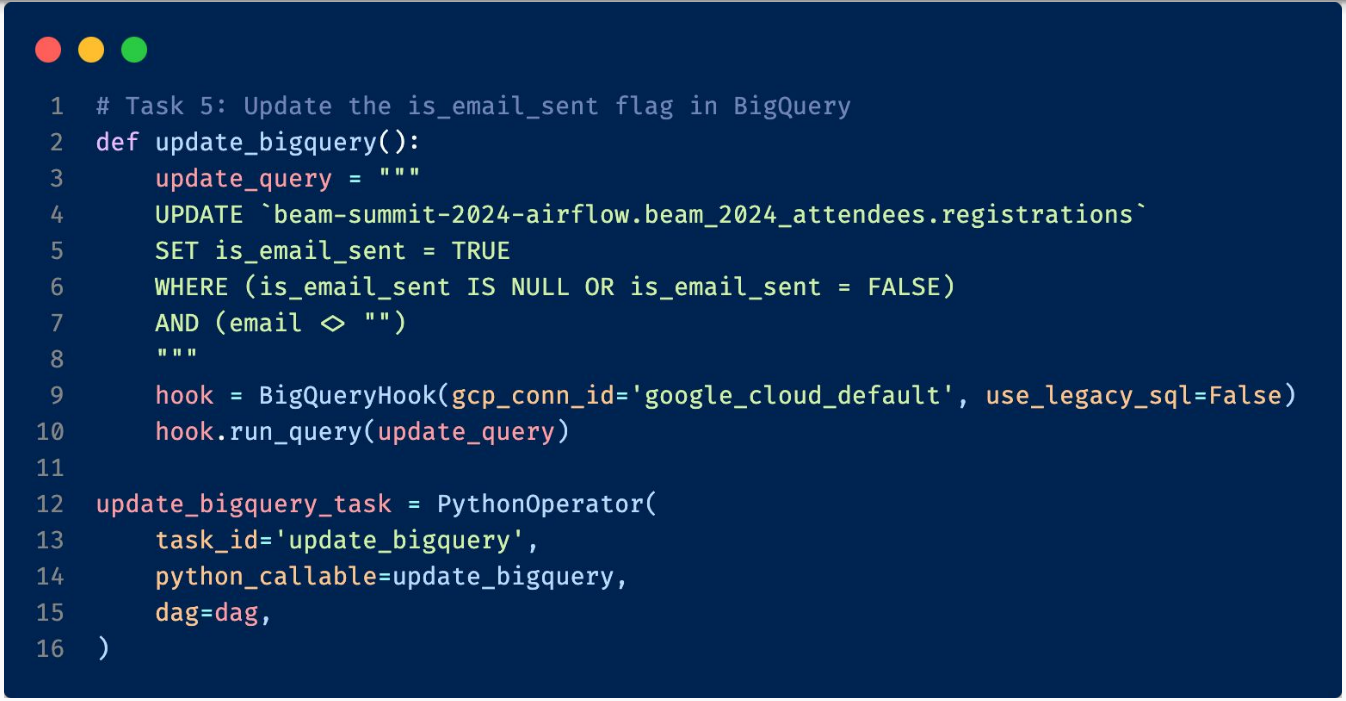
Sending Emails in Airflow using SendGrid

- **Create an account at** https://sendgrid.com/ and register your API key in the Airflow Environment Variable.

- **Call the Generative AI Model** `describe_this_location(location)` function to generate fun facts about the location entered in the csv file earlier.

- Infuse the results returned with a preconfigured text to for the email body. The use the Mail function to send the email.

Update the Email Flag column in BigQuery from Airflow

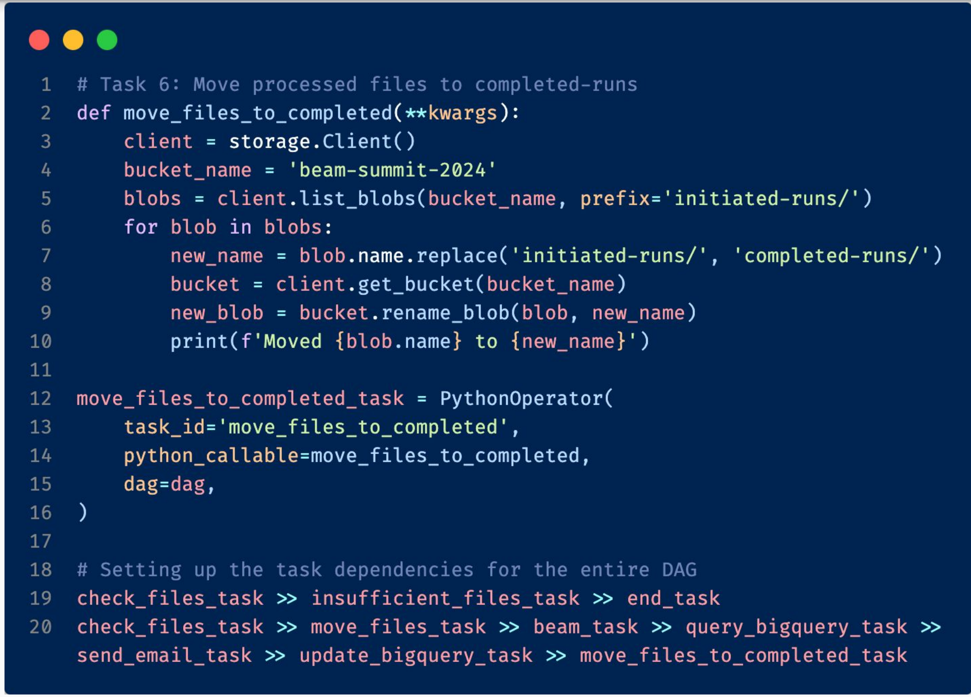


```python
# Task 5: Update the is_email_sent flag in BigQuery
def update_bigquery():
    update_query = """
    UPDATE `beam-summit-2024-airflow.beam_2024_attendees.registrations`
    SET is_email_sent = TRUE
    WHERE (is_email_sent IS NULL OR is_email_sent = FALSE)
    AND (email <> "")
    """
    hook = BigQueryHook(gcp_conn_id='google_cloud_default', use_legacy_sql=False)
    hook.run_query(update_query)

update_bigquery_task = PythonOperator(
    task_id='update_bigquery',
    python_callable=update_bigquery,
    dag=dag,
)
```

BEAM SUMMIT

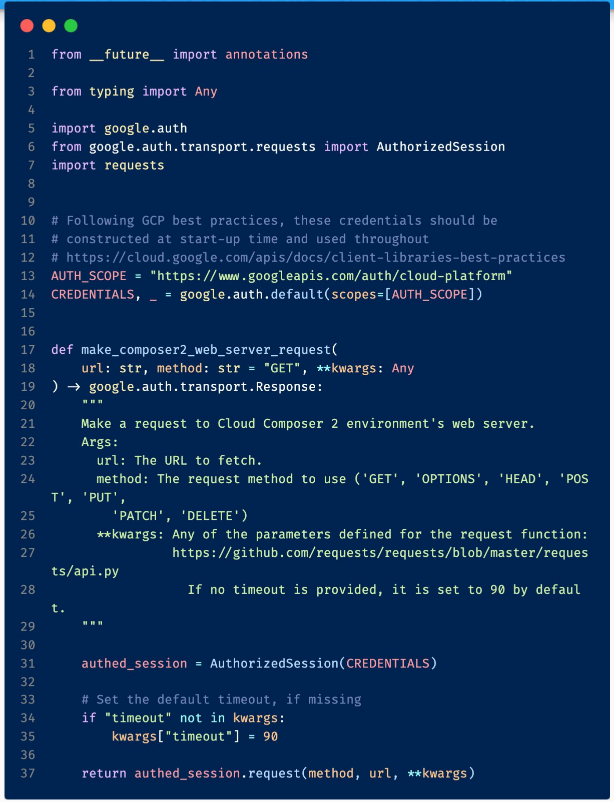
Final Steps: Setup the Task Dependencies

- **DAG Structure:** Tasks are organized in a DAG, with nodes representing tasks and edges representing dependencies.
- **Upstream/Downstream:** Tasks must complete upstream tasks before downstream tasks can start.
- **Setting Dependencies:** Use **>>, <<, *set_upstream()***, and ***set_downstream()*** to define task order.
- **Trigger Rules:** Control task execution based on upstream task outcomes (e.g., all_success, one_success).
- **Cross-DAG Dependencies**: Enable tasks in one DAG to trigger tasks in another using sensors.
- **Task Groups:** Group tasks for better management and visualization of dependencies.

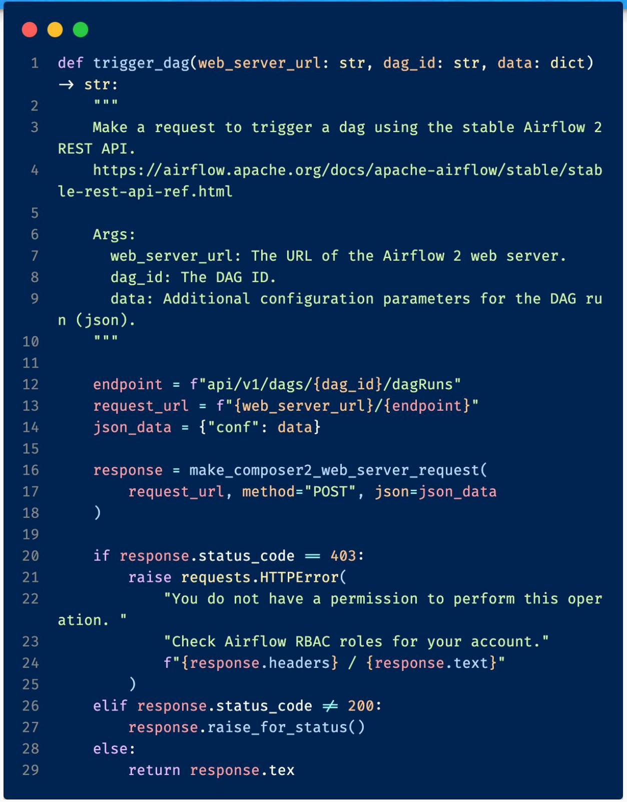


```python
# Task 6: Move processed files to completed-runs
def move_files_to_completed(**kwargs):
    client = storage.Client()
    bucket_name = 'beam-summit-2024'
    blobs = client.list_blobs(bucket_name, prefix='initiated-runs/')
    for blob in blobs:
        new_name = blob.name.replace('initiated-runs/', 'completed-runs/')
        bucket = client.get_bucket(bucket_name)
        new_blob = bucket.rename_blob(blob, new_name)
        print(f'Moved {blob.name} to {new_name}')

move_files_to_completed_task = PythonOperator(
    task_id='move_files_to_completed',
    python_callable=move_files_to_completed,
    dag=dag,
)

# Setting up the task dependencies for the entire DAG
check_files_task >> insufficient_files_task >> end_task
check_files_task >> move_files_task >> beam_task >> query_bigquery_task >>
send_email_task >> update_bigquery_task >> move_files_to_completed_task
```

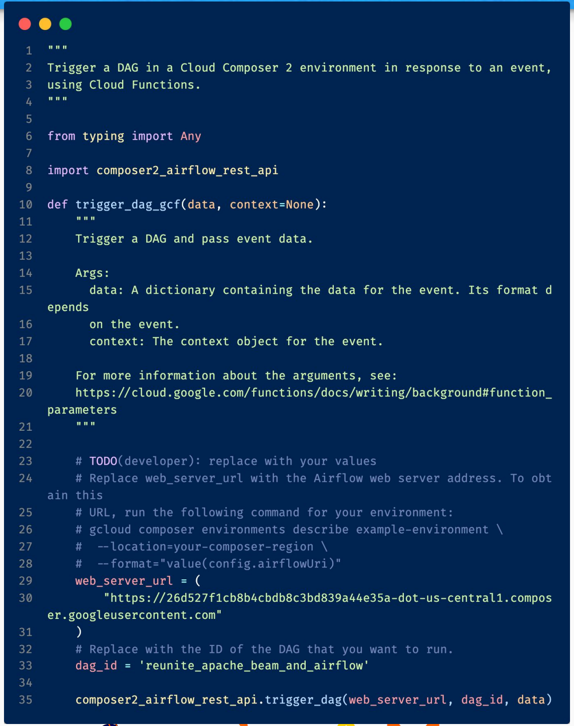
Use Cloud Functions to Trigger the Airflow DAG once there’s a new *.csv file in GCS



```python
from __future__ import annotations

from typing import Any

import google.auth
from google.auth.transport.requests import AuthorizedSession
import requests


# Following GCP best practices, these credentials should be
# constructed at start-up time and used throughout
# https://cloud.google.com/apis/docs/client-libraries-best-practices
AUTH_SCOPE = "https://www.googleapis.com/auth/cloud-platform"
CREDENTIALS, _ = google.auth.default(scopes=[AUTH_SCOPE])


def make_composer2_web_server_request(
    url: str, method: str = "GET", **kwargs: Any
) -> google.auth.transport.Response:
    """
    Make a request to Cloud Composer 2 environment's web server.
    Args:
      url: The URL to fetch.
      method: The request method to use ('GET', 'OPTIONS', 'HEAD', 'POST', 'PUT',
        'PATCH', 'DELETE')
      **kwargs: Any of the parameters defined for the request function:
                https://github.com/requests/requests/blob/master/requests/api.py
                  If no timeout is provided, it is set to 90 by default.
    """

    authed_session = AuthorizedSession(CREDENTIALS)

    # Set the default timeout, if missing
    if "timeout" not in kwargs:
        kwargs["timeout"] = 90

    return authed_session.request(method, url, **kwargs)
```

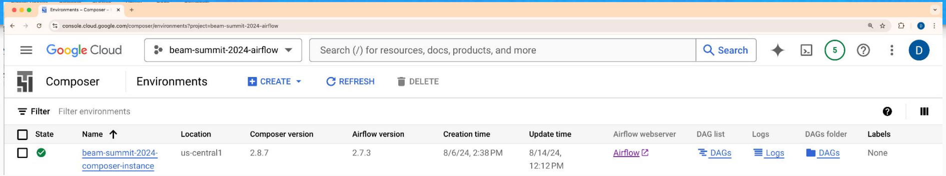


```python
def trigger_dag(web_server_url: str, dag_id: str, data: dict) -> str:
    """
    Make a request to trigger a dag using the stable Airflow 2 REST API.
    https://airflow.apache.org/docs/apache-airflow/stable/stable-rest-api-ref.html

    Args:
      web_server_url: The URL of the Airflow 2 web server.
      dag_id: The DAG ID.
      data: Additional configuration parameters for the DAG run (json).
    """

    endpoint = f"api/v1/dags/{dag_id}/dagRuns"
    request_url = f"{web_server_url}/{endpoint}"
    json_data = {"conf": data}

    response = make_composer2_web_server_request(
        request_url, method="POST", json=json_data
    )

    if response.status_code == 403:
        raise requests.HTTPError(
            "You do not have a permission to perform this operation. "
            "Check Airflow RBAC roles for your account."
            f"{response.headers} / {response.text}"
        )
    elif response.status_code != 200:
        response.raise_for_status()
    else:
        return response.tex
```

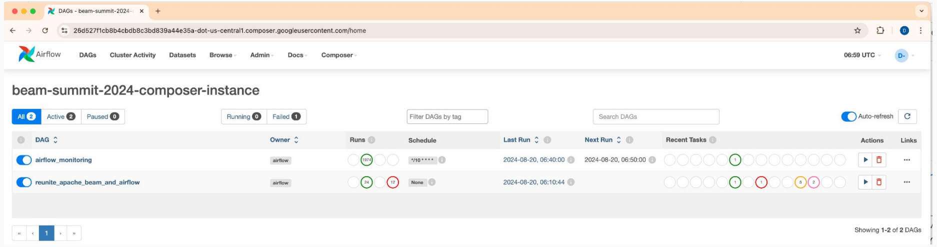


```python
"""
Trigger a DAG in a Cloud Composer 2 environment in response to an event,
using Cloud Functions.
"""

from typing import Any

import composer2_airflow_rest_api

def trigger_dag_gcf(data, context=None):
    """
    Trigger a DAG and pass event data.

    Args:
      data: A dictionary containing the data for the event. Its format depends
      on the event.
      context: The context object for the event.

    For more information about the arguments, see:
    https://cloud.google.com/functions/docs/writing/background#function_parameters
    """

    # TODO(developer): replace with your values
    # Replace web_server_url with the Airflow web server address. To obtain this
    # URL, run the following command for your environment:
    # gcloud composer environments describe example-environment \
    #  --location=your-composer-region \
    #  --format="value(config.airflowUri)"
    web_server_url = (
        "https://26d527f1cb8b4cbdb8c3bd839a44e35a-dot-us-central1.composer.googleusercontent.com"
    )
    # Replace with the ID of the DAG that you want to run.
    dag_id = 'reunite_apache_beam_and_airflow'

    composer2_airflow_rest_api.trigger_dag(web_server_url, dag_id, data)
```

Results (Airflow and Email)


Environments – Composer
console.cloud.google.com/composer/environments?project=beam-summit-2024-airflow
Google Cloud
beam-summit-2024-airflow
Search (/) for resources, docs, products, and more
Search
Composer
Environments
CREATE
REFRESH
DELETE
Filter
Filter environments

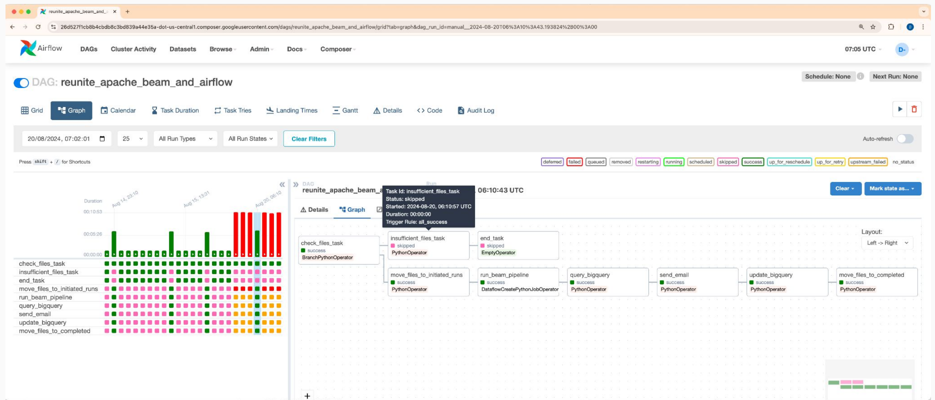
State	Name	Location	Composer version	Airflow version	Creation time	Update time	Airflow webserver	DAG list	Logs	DAGs folder	Labels
✓	beam-summit-2024-composer-instance	us-central1	2.8.7	2.7.3	8/6/24, 2:38 PM	8/14/24, 12:12 PM	Airflow	DAGs	Logs	DAGs	None



beam-summit-2024-composer-instance

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
airflow_monitoring	airflow	1974	*/10 * * * *	2024-08-20, 06:40:00	2024-08-20, 06:50:00	1		
reunite_apache_beam_and_airflow	airflow	24, 12	None	2024-08-20, 06:10:44		1, 1, 5, 2		

Showing 1-2 of 2 DAGs

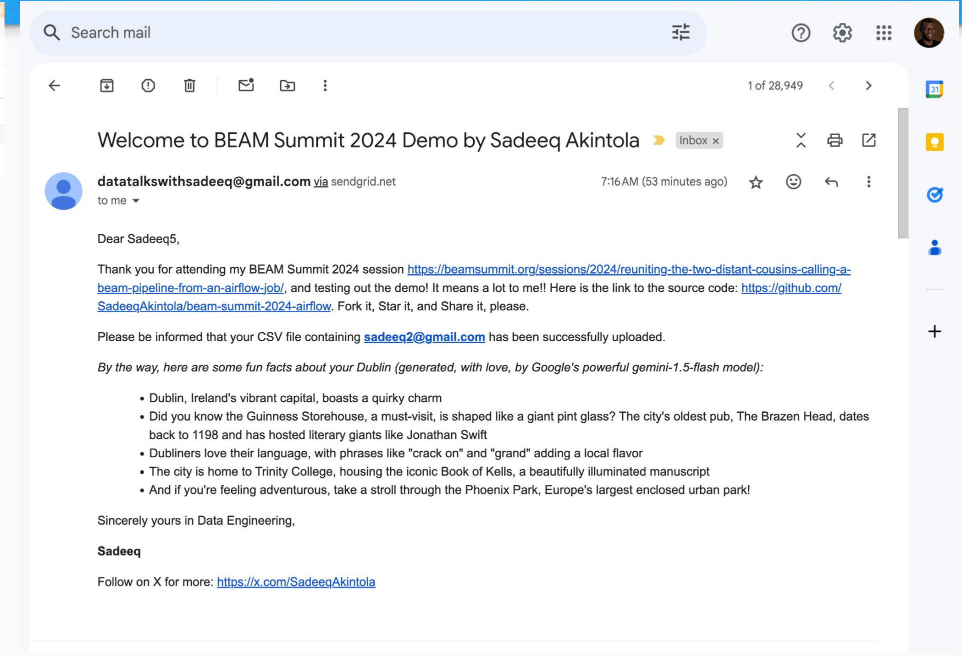


DAG: reunite_apache_beam_and_airflow

Run 06:10:43 UTC

Task Id: insufficient_files_task
Status: skipped
Started: 2024-08-20, 06:10:57 UTC
Duration: 00:00:00
Trigger Rule: all_success

check_files_task — success — BranchPythonOperator
insufficient_files_task — skipped — PythonOperator
end_task — skipped — EmptyOperator
move_files_to_initiated_runs — success — PythonOperator
run_beam_pipeline — success — DataflowCreatePythonJobOperator
query_bigquery — success — PythonOperator
send_email — success — PythonOperator
update_bigquery — success — PythonOperator
move_files_to_completed — success — PythonOperator



Welcome to BEAM Summit 2024 Demo by Sadeeq Akintola

datatalkswithsadeeq@gmail.com via sendgrid.net — 7:16 AM (53 minutes ago)

Dear Sadeeq5,

Thank you for attending my BEAM Summit 2024 session https://beamsummit.org/sessions/2024/reuniting-the-two-distant-cousins-calling-a-beam-pipeline-from-an-airflow-job/, and testing out the demo! It means a lot to me!! Here is the link to the source code: https://github.com/SadeeqAkintola/beam-summit-2024-airflow. Fork it, Star it, and Share it, please.

Please be informed that your CSV file containing **sadeeq2@gmail.com** has been successfully uploaded.

By the way, here are some fun facts about your Dublin (generated, with love, by Google's powerful gemini-1.5-flash model):

- Dublin, Ireland's vibrant capital, boasts a quirky charm
- Did you know the Guinness Storehouse, a must-visit, is shaped like a giant pint glass? The city's oldest pub, The Brazen Head, dates back to 1198 and has hosted literary giants like Jonathan Swift
- Dubliners love their language, with phrases like "crack on" and "grand" adding a local flavor
- The city is home to Trinity College, housing the iconic Book of Kells, a beautifully illuminated manuscript
- And if you're feeling adventurous, take a stroll through the Phoenix Park, Europe's largest enclosed urban park!

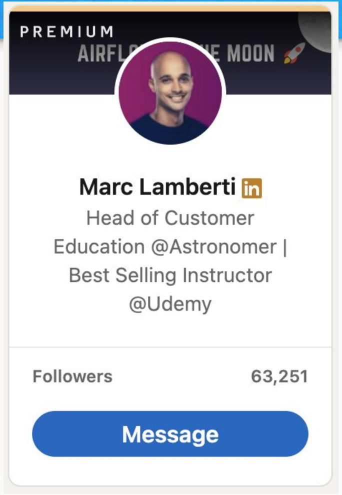
Sincerely yours in Data Engineering,

Sadeeq

Follow on X for more: https://x.com/SadeeqAkintola

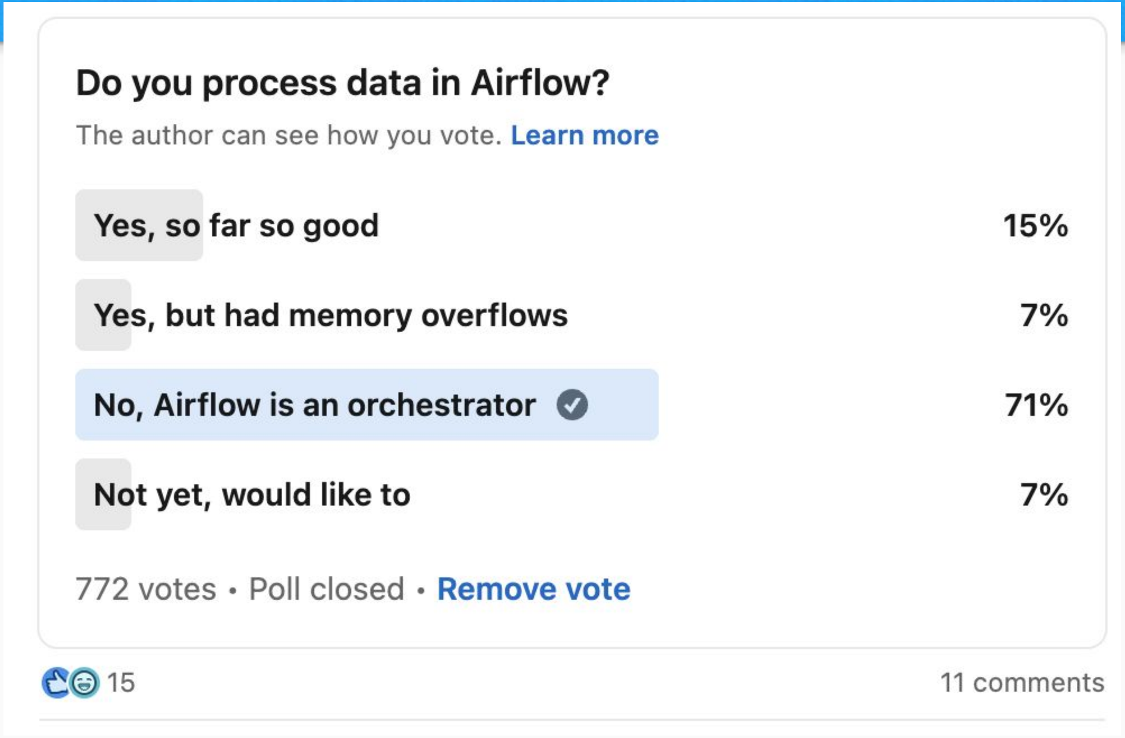
BEAM SUMMIT

Remember: “Airflow is (just) an Orchestrator”



Marc Lamberti
Head of Customer Education @Astronomer | Best Selling Instructor @Udemy

Followers 63,251



Do you process data in Airflow?

The author can see how you vote. Learn more

Yes, so far so good	15%
Yes, but had memory overflows	7%
No, Airflow is an orchestrator ✓	71%
Not yet, would like to	7%

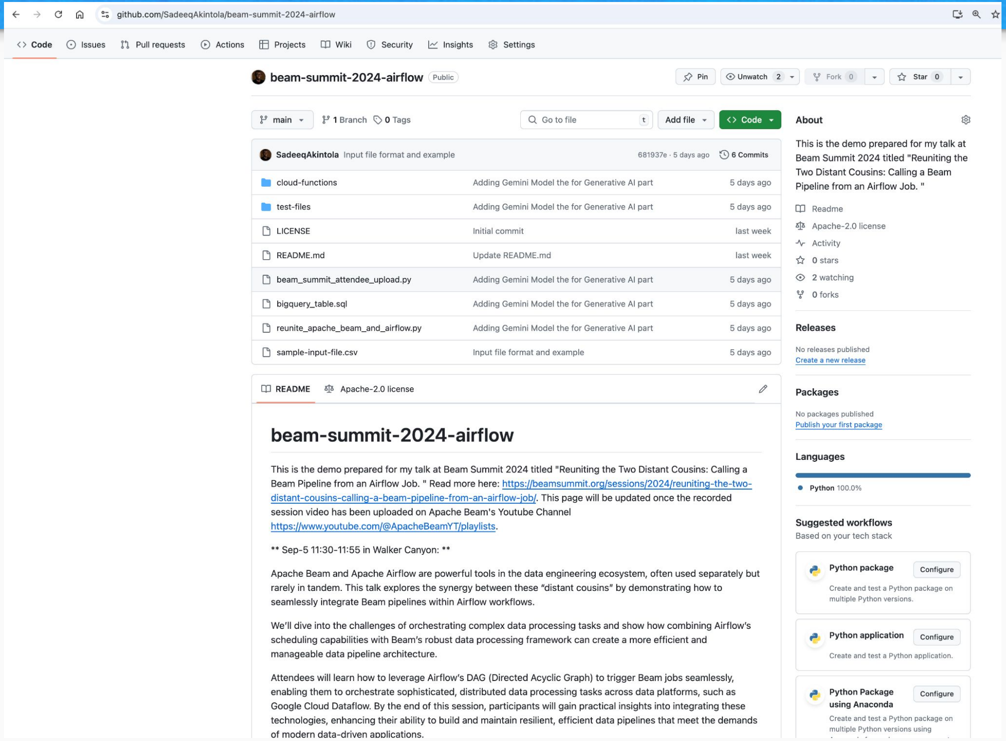
772 votes • Poll closed • Remove vote

15 · 11 comments

Demo!

BEAM SUMMIT

Have you tried the Demo? Starred/Cloned the Repo? Oya, Do it now!!



github.com/SadeeqAkintola/beam-summit-2024-airflow

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings

beam-summit-2024-airflow Public

Pin · Unwatch 2 · Fork 0 · Star 0

main · 1 Branch · 0 Tags · Go to file · Add file · Code

SadeeqAkintola Input file format and example 681937e · 5 days ago 6 Commits

cloud-functions	Adding Gemini Model the for Generative AI part	5 days ago
test-files	Adding Gemini Model the for Generative AI part	5 days ago
LICENSE	Initial commit	last week
README.md	Update README.md	last week
beam_summit_attendee_upload.py	Adding Gemini Model the for Generative AI part	5 days ago
bigquery_table.sql	Adding Gemini Model the for Generative AI part	5 days ago
reunite_apache_beam_and_airflow.py	Adding Gemini Model the for Generative AI part	5 days ago
sample-input-file.csv	Input file format and example	5 days ago

README · Apache-2.0 license

beam-summit-2024-airflow

This is the demo prepared for my talk at Beam Summit 2024 titled "Reuniting the Two Distant Cousins: Calling a Beam Pipeline from an Airflow Job. " Read more here: https://beamsummit.org/sessions/2024/reuniting-the-two-distant-cousins-calling-a-beam-pipeline-from-an-airflow-job/. This page will be updated once the recorded session video has been uploaded on Apache Beam's Youtube Channel https://www.youtube.com/@ApacheBeamYT/playlists.

** Sep-5 11:30-11:55 in Walker Canyon: **

Apache Beam and Apache Airflow are powerful tools in the data engineering ecosystem, often used separately but rarely in tandem. This talk explores the synergy between these "distant cousins" by demonstrating how to seamlessly integrate Beam pipelines within Airflow workflows.

We'll dive into the challenges of orchestrating complex data processing tasks and show how combining Airflow's scheduling capabilities with Beam's robust data processing framework can create a more efficient and manageable data pipeline architecture.

Attendees will learn how to leverage Airflow's DAG (Directed Acyclic Graph) to trigger Beam jobs seamlessly, enabling them to orchestrate sophisticated, distributed data processing tasks across data platforms, such as Google Cloud Dataflow. By the end of this session, participants will gain practical insights into integrating these technologies, enhancing their ability to build and maintain resilient, efficient data pipelines that meet the demands of modern data-driven applications.

About

This is the demo prepared for my talk at Beam Summit 2024 titled "Reuniting the Two Distant Cousins: Calling a Beam Pipeline from an Airflow Job. "

Readme · Apache-2.0 license · Activity · 0 stars · 2 watching · 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

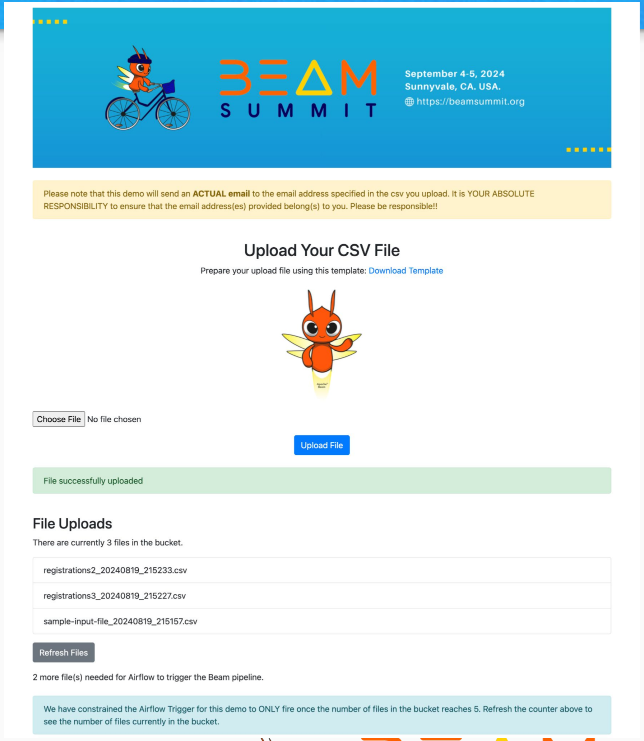
● Python 100.0%

Suggested workflows
Based on your tech stack

Python package — Configure
Create and test a Python package on multiple Python versions.

Python application — Configure
Create and test a Python application.

Python Package using Anaconda — Configure
Create and test a Python package on multiple Python versions using



Please note that this demo will send an **ACTUAL email** to the email address specified in the csv you upload. It is YOUR ABSOLUTE RESPONSIBILITY to ensure that the email address(es) provided belong(s) to you. Please be responsible!!

Upload Your CSV File

Prepare your upload file using this template: Download Template

Choose File No file chosen

Upload File

File successfully uploaded

File Uploads

There are currently 3 files in the bucket.

registrations2_20240819_215233.csv
registrations3_20240819_215227.csv
sample-input-file_20240819_215157.csv

Refresh Files

2 more file(s) needed for Airflow to trigger the Beam pipeline.

We have constrained the Airflow Trigger for this demo to ONLY fire once the number of files in the bucket reaches 5. Refresh the counter above to see the number of files currently in the bucket.

Visit http://35.222.154.240:8080/ or https://SadeeqAkintola.com/ to try the demo.
Please Star, Fork and Clone the Code Repo here: https://github.com/SadeeqAkintola/beam-summit-2024-airflow

Useful Resources

Official Documentation:

1. **Apache Beam Documentation:** https://beam.apache.org/documentation/
2. **Apache Airflow Documentation:** https://airflow.apache.org/docs/
3. **Google Dataflow Documentation:** https://cloud.google.com/dataflow/docs/
4. **Triggering Beam Pipelines with Cloud Composer (Google Documentation):** https://cloud.google.com/composer/docs/how-to/using/triggering-with-gcf

Popular Medium Posts:

5. **Event-Based Dataflow Job Orchestration with Cloud Composer, Airflow, and Cloud Functions:** https://qulia.medium.com/event-based-dataflow-job-orchestration-with-cloud-composer-airflow-and-cloud-functions-b61219f9aeaf
6. **Launching Dataflow Pipelines via Cloud Composer (Airflow):** https://medium.com/@kolban1/cloud-composer-launching-dataflow-pipelines-38cd29e970d4
7. **Launch an Apache Beam Pipeline with Apache Airflow — Part 1/2: Setting up the Airflow Environment with Docker-Compose:** https://medium.com/@carmelwenga/launch-an-apache-beam-pipeline-with-apache-airflow-part-1-setting-up-the-airflow-environment-d97dd64ded18

YouTube Videos:

8. **Apache Beam: A Unified Model for Batch and Streaming Data Processing:** https://www.youtube.com/watch?v=7DZ8ONmeP5A
9. **Flexible, Easy Data Pipelines on Google Cloud with Cloud Composer (Cloud Next '18):** https://www.youtube.com/watch?v=GeNFEtt-D4k
10. **Cloud Composer - Orchestrating an ETL Pipeline Using Cloud Dataflow:** https://www.youtube.com/watch?v=PCg9AQNuK3E

Also, join us for Airflow Summit Next Week: https://airflowsummit.org

BEAM SUMMIT

Interested in learning more about Cloud, Data and AI?

Schedule a conversation with me:

X / Twitter: @SadeeqAkintola
Online: SadeeqAkintola.com

BEAM SUMMIT