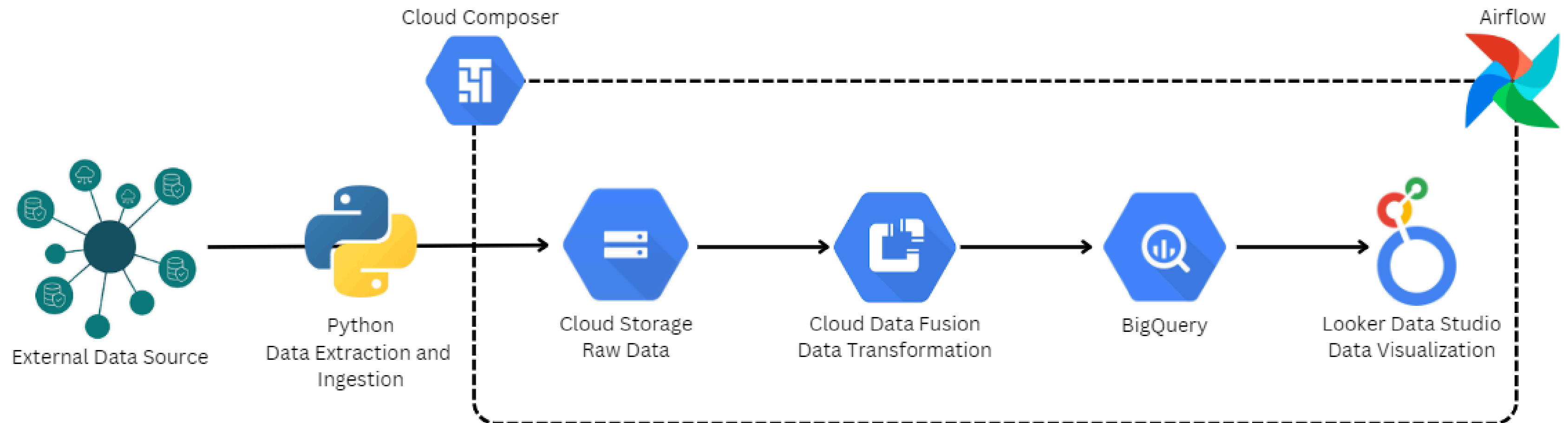


ETL Pipeline and Reporting in Google Cloud



Problem Statement:

You are tasked with designing and implementing a comprehensive data pipeline to manage employee data from multiple sources, ensuring that sensitive information is handled securely and appropriately masked. This data will then be loaded into Google BigQuery, where it will be available for secure visualization on a web-based dashboard.



Requirements

Data Extraction:

- Extract employee data from various sources, including databases, CSV files, and APIs.
- Ensure the process is efficient, scalable, and can handle data from multiple formats and sources.

Data Masking:

- Identify and mask sensitive information within the employee data, such as social security numbers, salary details, and personal contact information.
- Implement robust data masking techniques to maintain data privacy and security.

Data Loading into BigQuery:

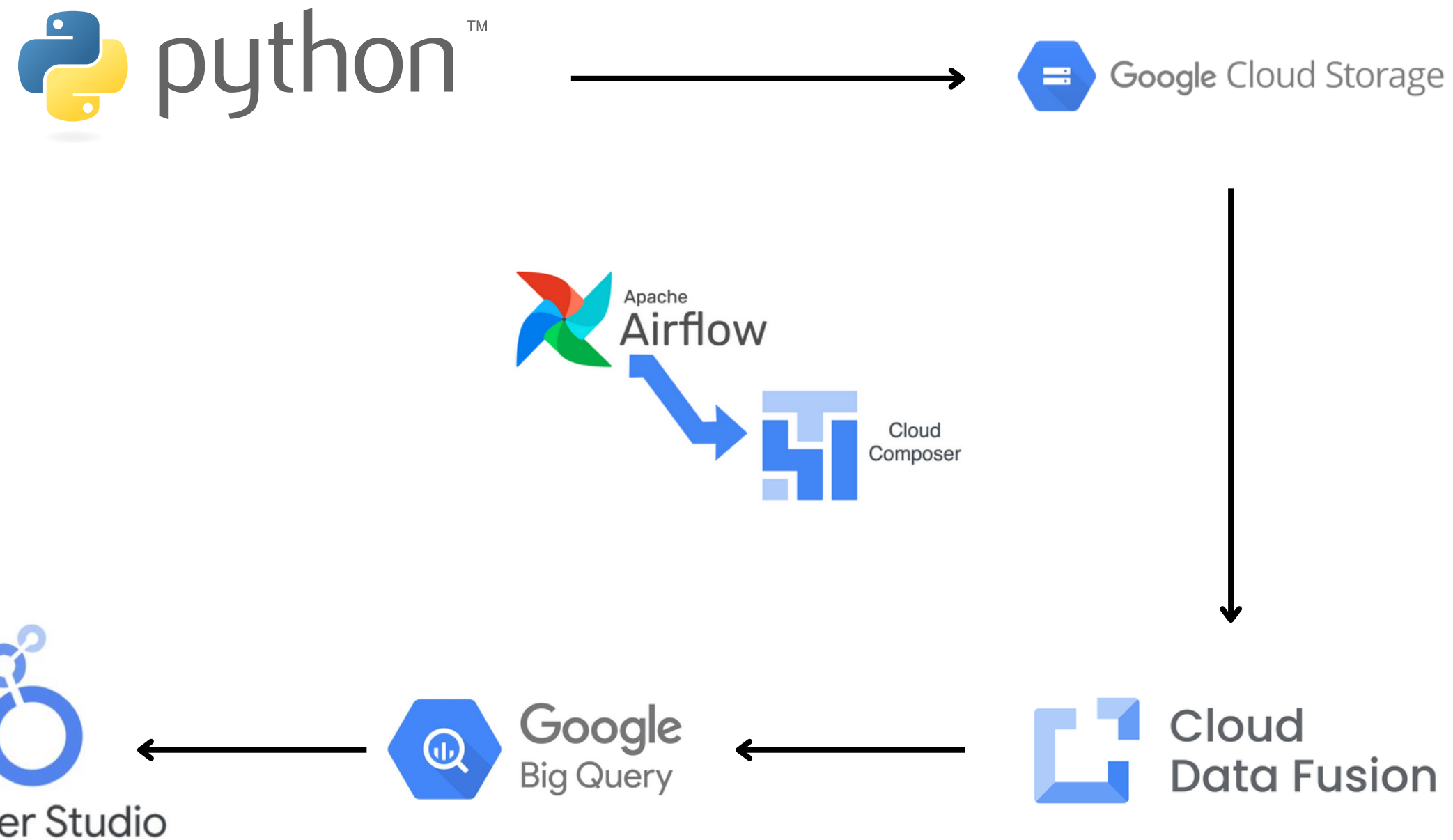
- Design and implement a secure process to load the masked employee data into Google BigQuery.
- Ensure the data loading process is optimized for performance, reliability, and scalability.

Dashboard Visualization:

- Develop a user-friendly, web-based dashboard using visualization tools such as Google Data Studio, Tableau, or custom dashboards.
- Ensure that the dashboard provides clear, secure, and insightful visualizations of the employee data.

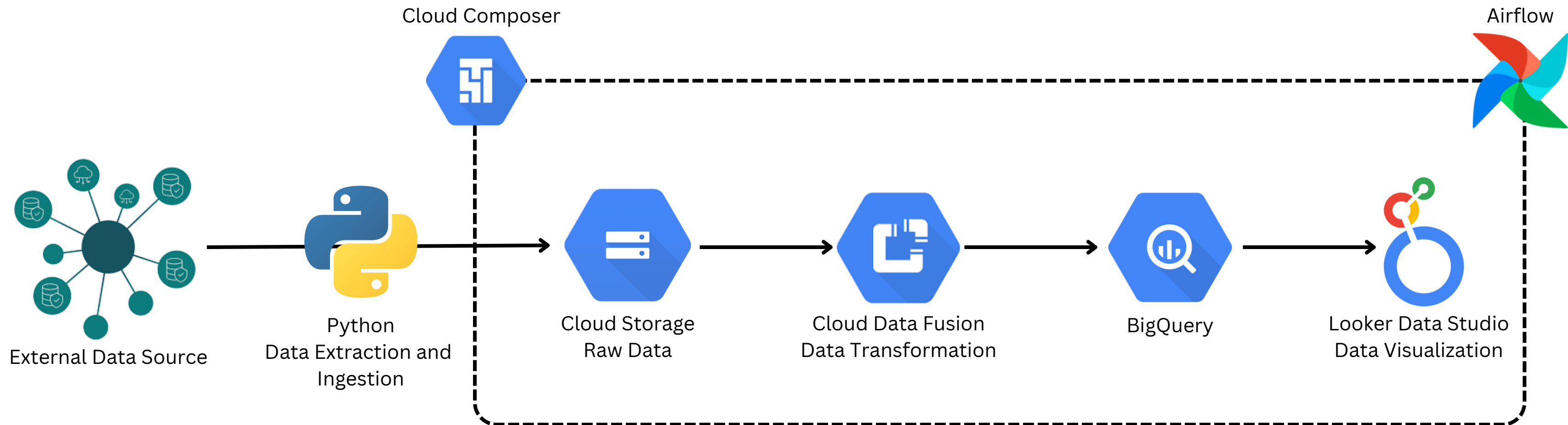


TechStack



1. Python will be used for Data Extraction
2. We will use google cloud storage for storing the RAW data
3. Cloud data fusion as a data pipeline where we will be doing data transformation
4. Then we will load the data in BigQuery and visualize the data in looker studio.
5. Finally all the above process will be orchestrating on cloud composer using Airflow.
6. At the end we will visualize it in the looker studio

Pipeline: From Extraction to Visualization



Above pipeline shows data flows from external data sources into a system for extraction and ingestion using Python. The ingested raw data is stored in Google Cloud Storage, from where it is processed and transformed using Cloud Data Fusion. The transformed data is then loaded into BigQuery for data warehousing and further analysis. Finally, the data is visualized using Looker Data Studio (Google Data Studio). The entire pipeline is orchestrated by Cloud Composer, utilizing Apache Airflow to automate and schedule tasks.

Creating a Composer

Google Cloud

data-pipeline

Search (/) for resources, docs, products, and more

Search

Composer

Environments

+

CREATE

REFRESH

DELETE

Filter

Filter environments

<div><input type="checkbox"/></div>	State	Name <div>↑</div>	Location	Composer version	Airflow version	Creation time	Update time	Airflow webserver	DAG list	Logs	DAGs folder	Labels
<div><input type="checkbox"/></div>	<div></div>	composer-dev	europe-central2	3	2.9.3-build.0	9/22/24, 11:23 AM	9/22/24, 11:23 AM	None	<div><div></div><div>DAGs</div></div>	<div><div></div><div>Logs</div></div>	None	None

Here I have created cloud composer '***composer-dev***' which helps to create managed Airflow environments quickly and use Airflow-native tools

Google Cloud Composer is a workflow orchestration service that helps users:

- **Create, schedule, and monitor workflows:** Build, schedule, and monitor workflows across clouds and on-premises data centers
- **Integrate with other Google products:** Connect with other Google Cloud products like BigQuery, Dataflow, and Cloud Storage
- **Use Airflow-native tools:** Use Airflow's web interface and command-line tools
- **Automate complex workflows:** Streamline and automate workflows for data processing, machine learning, DevOps automation, and business process management

Creating a Data Fusion

Google Cloud

data-pipeline

Search (/) for resources, docs, products, and more

Search

3

Data Fusion

Instances

Permissions

Instances

CREATE INSTANCE

REFRESH

DELETE

UPGRADE

How was your experience?

Select which instance of Cloud Data Fusion you want to view.

		Instance Name	Action	Edition	Region	Zone	Version	Notifications	Encryption	Last Updated
<input type="checkbox"/>	●									
<input type="checkbox"/>	*	data-fusion-dev	Creating...	Basic	me-central2	--	6.10.1(6.10.1.1)		Google-managed	Sep 22, 2024, 11:30:11 AM

Created a data fusion ***'data-fusion-dev'***

- Google Cloud Data Fusion is a cloud-native data integration service that helps users build and manage data pipelines:
- Data integration: Users can connect to various data sources, transform the data, and transfer it to other systems.
 - Data lakes: Users can build scalable, distributed data lakes by integrating data from on-premises platforms.
 - Data warehouses: Users can create data warehouses in BigQuery.
 - Data visualization: Users can visualize data in Google Data Studio dashboards.

Creating a Bucket

Google Cloud

data-pipeline

Search (/) for resources, docs, products, and more

Search

4

B

Cloud Storage

Overview

Buckets

Monitoring

Settings

Buckets

CREATE

REFRESH

GO TO PATH

LEARN

Review the soft delete settings on your buckets. Billing for soft deleted objects will begin on September 1st.

LEARN MORE

MANAGE SOFT DELETE POLICIES

A new Cloud Storage overview page has been released. It will become the Cloud Storage landing page in October 2024.

TAKE A LOOK

Filter

Filter buckets

<input type="checkbox"/>	Name ↑	Created	Location type	Location	Default storage class ?	Last modified	Public access ?	
<input type="checkbox"/>	bucket-emp-data	Sep 22, 2024, 12:01:39 PM	Multi-region	us	Standard	Sep 22, 2024, 12:01:39 PM	Not public	

Created bucket ***'bucket-emp-data'*** to store employee data.

Google Cloud Storage is a service that allows users to store and retrieve data in Google Cloud. It offers a variety of storage options, including:

- Standard storage: For data that is accessed frequently, such as mobile apps, websites, and streaming videos
- Nearline storage: A low-cost, durable storage option for data that is accessed infrequently
- Coldline storage: A very low-cost, durable storage option for data that is accessed infrequently
- Archive storage: A low-cost, durable storage service for data archiving, disaster recovery, and online backup

Creating a Python Script

Created a python script .

Data Generation:

- Uses the Faker library to generate synthetic employee data, such as first name, last name, job title, department, email, address, phone number, and a random salary for 100 employees.
- Generates an 8-character password for each employee using a combination of letters, digits, and the character 'm'.

CSV File Creation:

- Opens a file named employee_data.csv and writes the employee data in CSV format, including headers for each field (e.g., first name, last name, job title).

Google Cloud Storage (GCS) Setup:

- Uses the google.cloud.storage library to create a client to connect to a Google Cloud Storage bucket.

Uploading the CSV File:

- Defines a function upload_to_gcs to upload the generated CSV file (employee_data.csv) to a specified GCS bucket (bucket-emp-data).
- Execution:
- Calls the function to upload the file, and upon success, prints a message confirming that the file has been uploaded to the desired bucket location in GCS.

```
1  import csv
2  from faker import Faker
3  import random
4  import string
5  from google.cloud import storage
6
7  # Specify number of employees to generate
8  num_employees = 100
9
10 # Create Faker instance
11 fake = Faker()
12
13 # Define the character set for the password
14 password_characters = string.ascii_letters + string.digits + 'm'
15
16 # Generate employee data and save it to a CSV file
17 with open('employee_data.csv', mode='w', newline='') as file:
18     fieldnames = ['first_name', 'last_name', 'job_title', 'department', 'email', 'address', 'phone_number', 'salary', 'password']
19     writer = csv.DictWriter(file, fieldnames=fieldnames)
20
21     writer.writeheader()
22     for _ in range(num_employees):
23         writer.writerow({
24             "first_name": fake.first_name(),
25             "last_name": fake.last_name(),
26             "job_title": fake.job(),
27             "department": fake.job(), # Generate department-like data using the job() method
28             "email": fake.email(),
29             "address": fake.city(),
30             "phone_number": fake.phone_number(),
31             "salary": fake.random_number(digits=5), # Generate a random 5-digit salary
32             "password": ''.join(random.choice(password_characters) for _ in range(8)) # Generate an 8-character password with 'm'
33         })
34
35 # Upload the CSV file to a GCS bucket
36 def upload_to_gcs(bucket_name, source_file_name, destination_blob_name):
37     storage_client = storage.Client()
38     bucket = storage_client.bucket(bucket_name)
39     blob = bucket.blob(destination_blob_name)
40
41     blob.upload_from_filename(source_file_name)
42
43     print(f'File {source_file_name} uploaded to {destination_blob_name} in {bucket_name}.')
44
45 # Set your GCS bucket name and destination file name
46 bucket_name = 'bucket-emp-data'
47 source_file_name = 'employee_data.csv'
48 destination_blob_name = 'employee_data.csv'
49
50 # Upload the CSV file to GCS
51 upload_to_gcs(bucket_name, source_file_name, destination_blob_name)
```

Data Fusion - Wrangler (Clean and Transform data)

Joining first_name and last_name as full_name

employee_data.csv

Cloud Storage Default - bucket-emp-data/emplo

employee_data.c

String	String
<input checked="" type="checkbox"/> first_name <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> last_name <input checked="" type="checkbox"/>
Lauren	Phelps
Elizabeth	Lee
Maria	Meza
Andrea	Rhodes
Stacy	Brown
Max	Davidson
Mariah	Thompson
Matthew	Green
Michael	Russo
Jennifer	Howard

Parse

Set character encoding

Change data type

Format

Calculate

Custom transform

Filter

Send to error

Find and replace

Fill null or empty cells

Copy column

Delete selected columns

Keep selected columns

Join two columns ☒

Swap two column names

Extract fields

Explode

Define variable

officer, museum

apist

ector

broadcasting

nce worker

Set order

first_name ☒

last_name ☒

Choose delimiter

Space ☒

Name new column

full_name| ☒

String

☒ full_name ☐

Lauren Phelps

Elizabeth Lee

Maria Meza

Andrea Rhodes

Stacy Brown

Max Davidson

Mariah Thompson

Matthew Green

Michael Russo

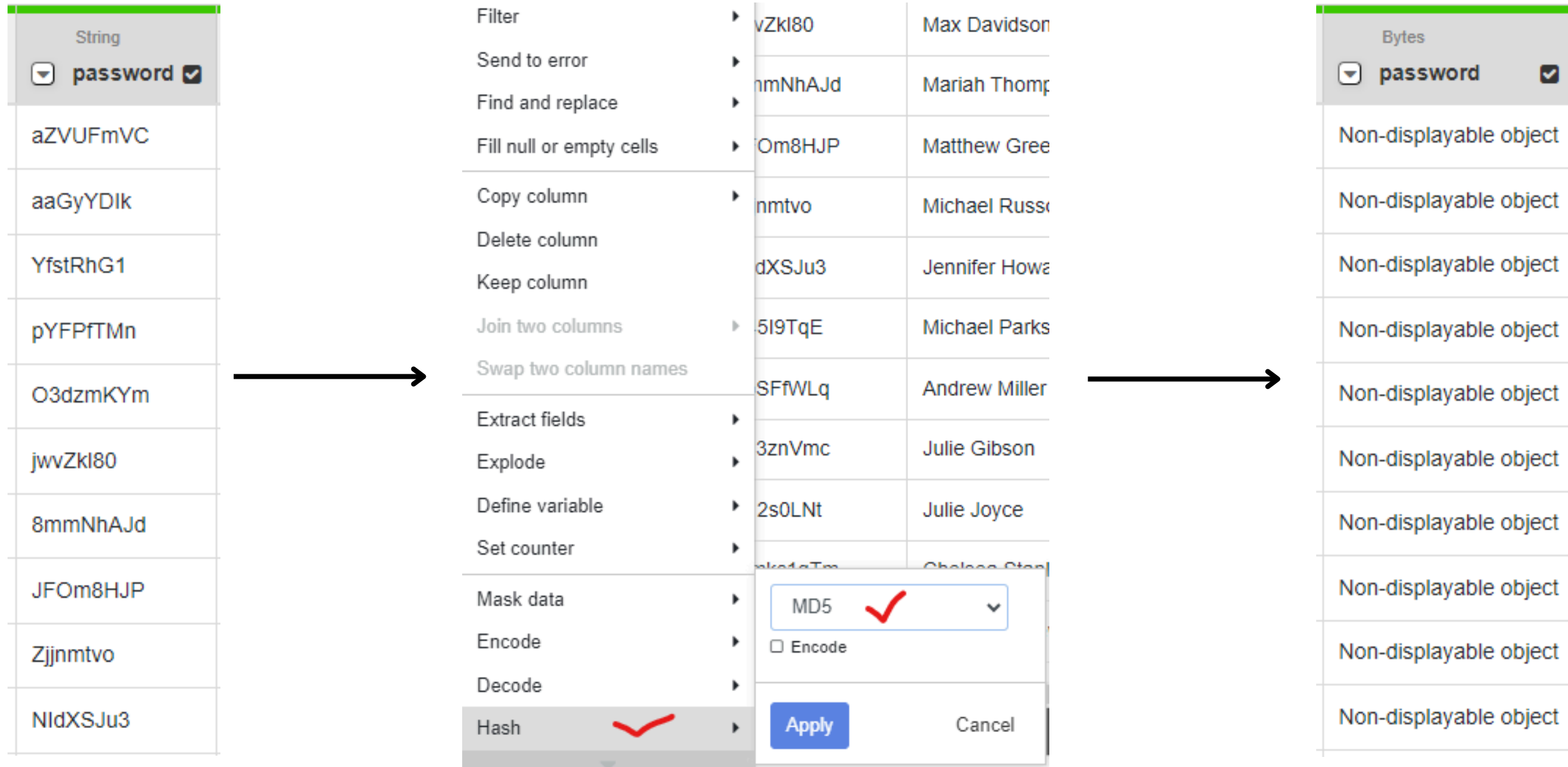
Jennifer Howard



Masking the salary column



Hashing the password column



cleaning and replacing email column






The diagram illustrates a three-step process for cleaning and replacing email addresses in a data column:

- Initial Data:** A column named 'email' containing various email addresses, all using the domain 'example.org' or 'example.net'.
- Transformation:** A context menu is open for the 'email' column. The 'Find and replace' option is selected. A sub-menu is displayed with the following settings:
 - Find: example
 - Exact match: ☒
 - Ignore case: ☐
 - Replace with: gmail
 - Buttons: Replace All, Cancel
- Result:** The 'email' column now contains the cleaned email addresses, where the domain has been replaced with 'gmail.com' or 'gmail.net'.

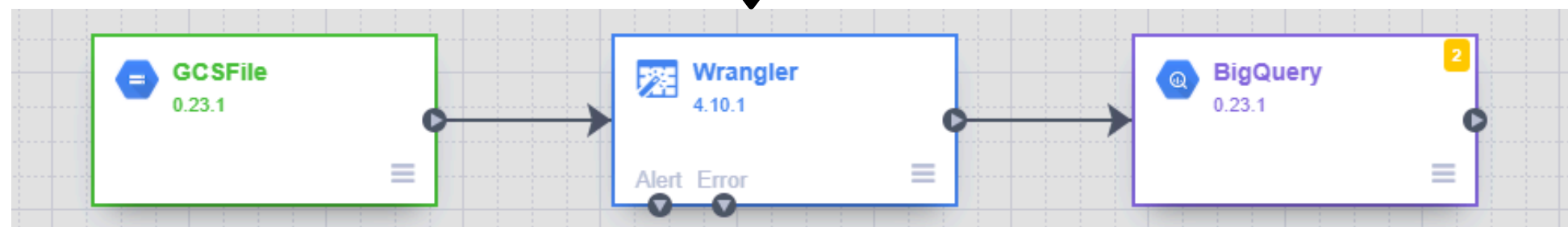
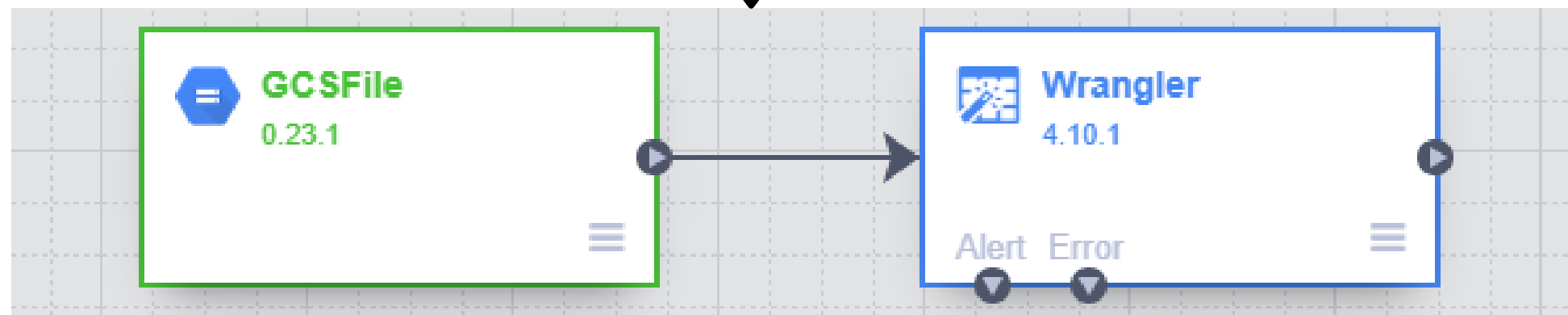
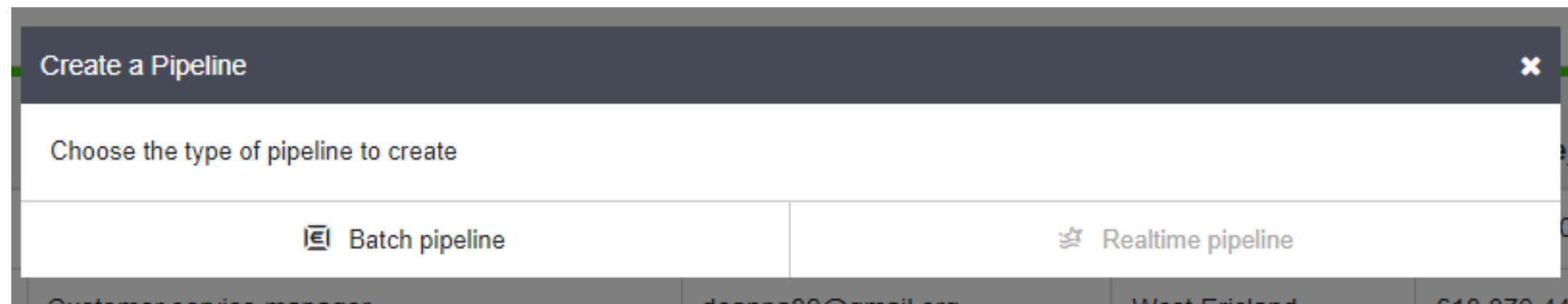
Initial Data	Transformation	Result
jaime55@example.com	Find and replace Find: example Exact match: <input checked="" type="checkbox"/> Ignore case: <input type="checkbox"/> Replace with: gmail Buttons: Replace All, Cancel	jaime55@gmail.com
deanna82@example.org		deanna82@gmail.org
andrew05@example.org		andrew05@gmail.org
scotttracey@example.com		scotttracey@gmail.com
donna79@example.org		donna79@gmail.org
jameslevine@example.org		jameslevine@gmail.org
marie09@example.org		marie09@gmail.org
breanna43@example.com		breanna43@gmail.com
pierceangela@example.net		pierceangela@gmail.net
johnnysnyder@example.org		johnnysnyder@gmail.org



Transformation steps

		Create a Pipeline	More
Columns (10)		Transformation steps (4)	
#	Transformations		
1	merge :first_name :last_name :full_name ''		
2	mask-number :salary xxxxx		
3	hash :password MD5 false		
4	find-and-replace :email s/example/gmail/g		

Creating Batch Pipeline



A batch pipeline is a system that processes large amounts of data in batches or at scheduled intervals. It's a structured and automated process that's often used for non-time-sensitive tasks and complex data processing.

At final, dumping the data into bigquery after transformation for analysis.

Setting up BigQuery services

Cloud Data Fusion | Studio

OPERATIONS

HUB

SYSTEM ADMIN

Basic Editor

BigQuery Properties 0.23.1

No errors found.

Validate

This sink writes to a BigQuery table. BigQuery is Google's serverless, highly scalable, enterprise data warehouse. Data is first written to a temporary location on Google Cloud Storage, then loaded into BigQuery from there.

Properties

Documentation

Input Schema

first_name	string	<div></div>	<div></div>	<div></div>
last_name	string	<div></div>	<div></div>	<div></div>
job_title	string	<div></div>	<div></div>	<div></div>
department	string	<div></div>	<div></div>	<div></div>
email	string	<div></div>	<div></div>	<div></div>
address	string	<div></div>	<div></div>	<div></div>
phone_number	string	<div></div>	<div></div>	<div></div>
salary	string	<div></div>	<div></div>	<div></div>
password	bytes	<div></div>	<div></div>	<div></div>
full_name	string	<div></div>	<div></div>	<div></div>

Use connection

NO

Project ID

auto-detect

Dataset Project ID

data-pipeline

Service Account Type

File Path JSON

Service Account File Path

auto-detect

Basic

Reference Name

pipeline-bigquery

BROWSE

Dataset *

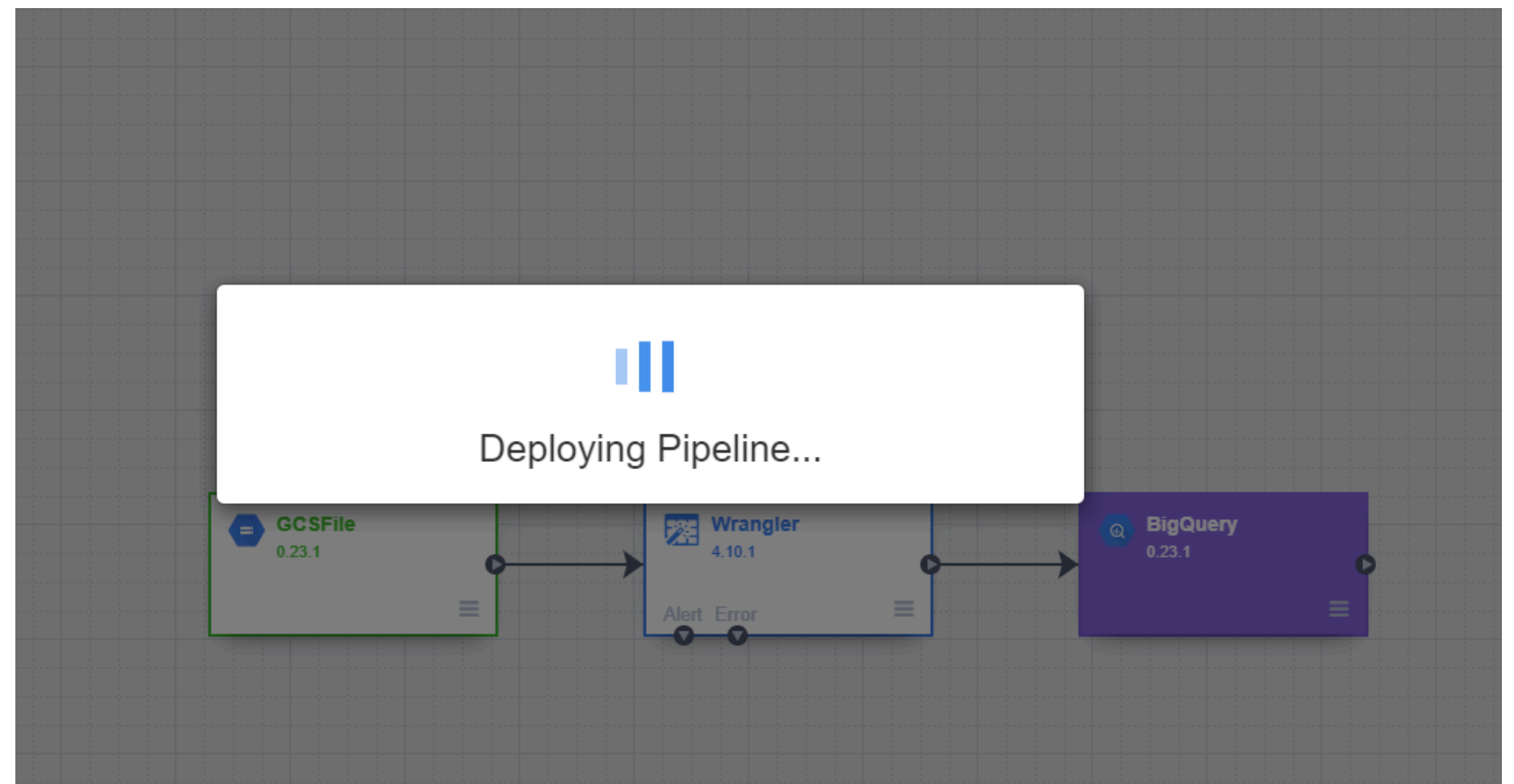
employee

Table *

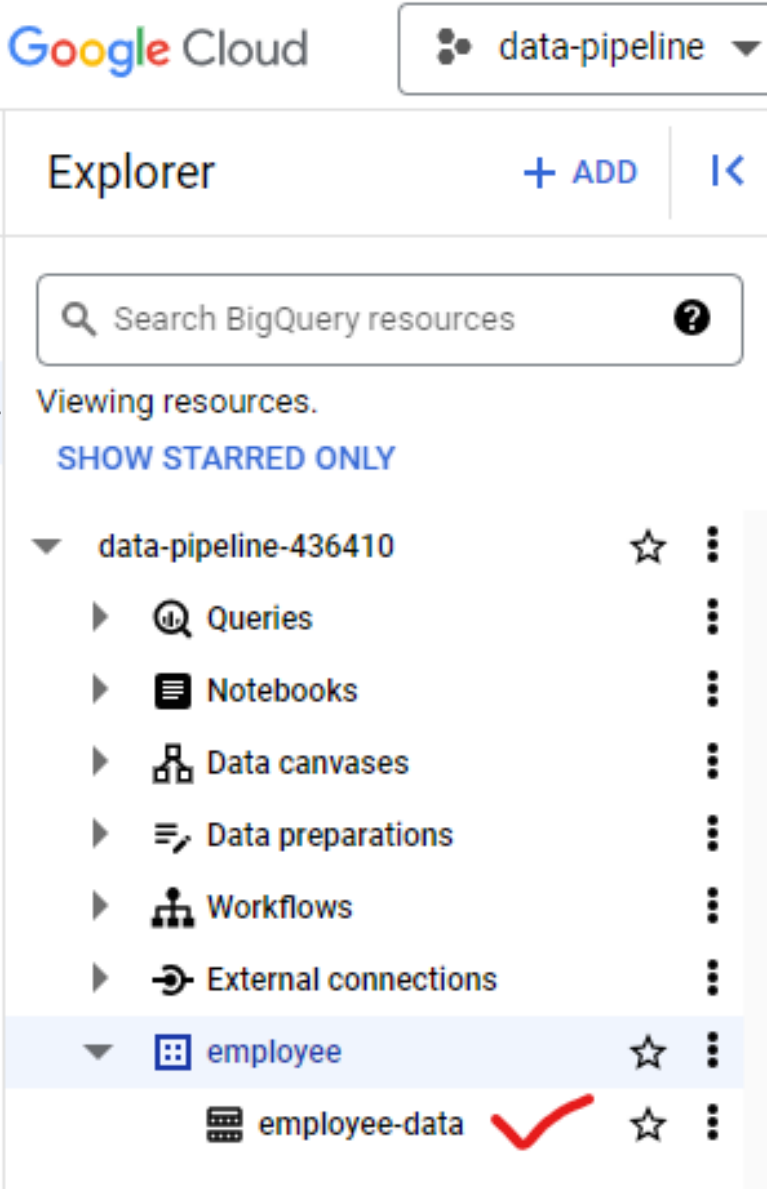
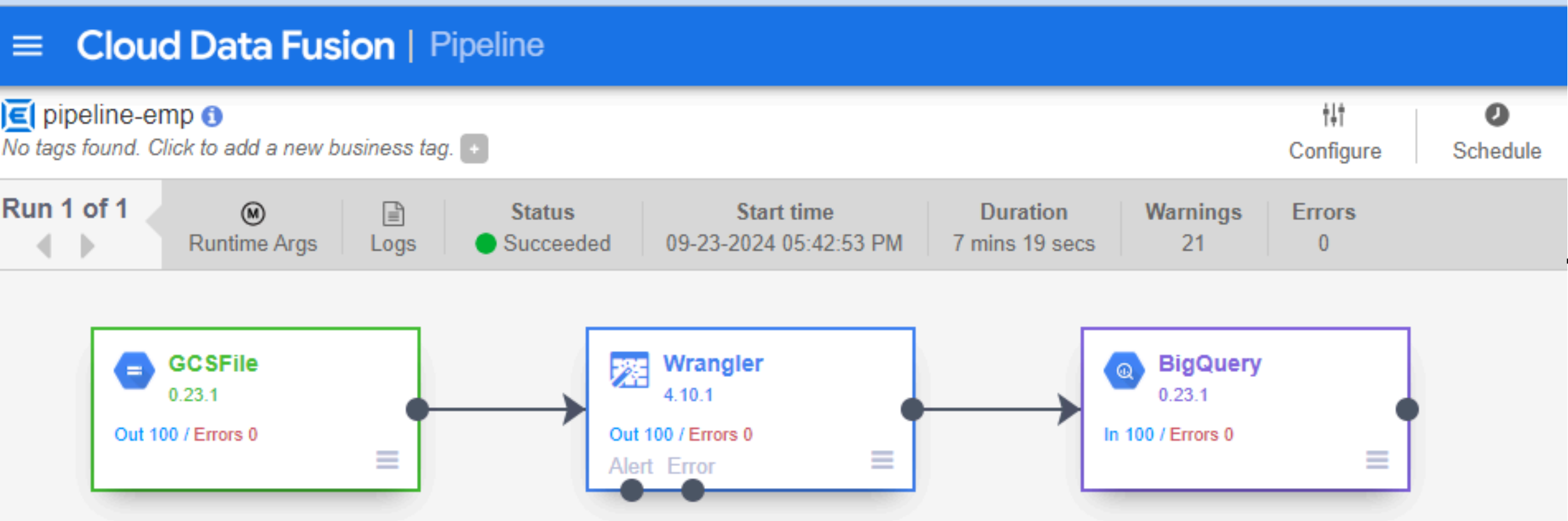
employee-data

Deploying Pipeline

After successfully creating pipeline and setting up my bigquery its time to deploy the data into the bigquery services for further analysis.



Data sinking into bigquery



As we can see employee-data is deployed into the bigquery services.

home

employee... ata

+

employee-data

QUERY

SHARE

COPY

This is a partitioned table. [Learn more](#)

SCHEMA

DETAILS

PREVIEW

TABLE EXPLORER

PREVIEW

Filter

Enter property name or value

	Field name	Type	Mode	Key	Collation
<input type="checkbox"/>	first_name	STRING	NULLABLE	-	-
<input type="checkbox"/>	last_name	STRING	NULLABLE	-	-
<input type="checkbox"/>	job_title	STRING	NULLABLE	-	-
<input type="checkbox"/>	department	STRING	NULLABLE	-	-
<input type="checkbox"/>	email	STRING	NULLABLE	-	-
<input type="checkbox"/>	address	STRING	NULLABLE	-	-
<input type="checkbox"/>	phone_number	STRING	NULLABLE	-	-
<input type="checkbox"/>	salary	STRING	NULLABLE	-	-
<input type="checkbox"/>	password	BYTES	NULLABLE	-	-
<input type="checkbox"/>	full_name	STRING	NULLABLE	-	-

Verifying Schema

Cross checking the deployed data into schema where I can check for columns with respective data type and data mode.

BigQuery to validate data transformation

```
SELECT * FROM `data-pipeline-436410.employee.employee-data` ;|
```

email ✓	address	phone_number	salary ✓	password ✓	full_name ✓
ashleyrice@gmail.com	Jeremymouth	564.456.5562x119	xxxxxx	hHbhmezgvmVQF3RFweHEtQ==	Ann Barnett
jameslevine@gmail.org	North Leslie	853.670.2963x040	xxxxxx	OaqRSx8xRaeyVRX5h3AOMg==	Max Davidson
jessica17@gmail.com	West Alexander	(205)603-7777	xxxxxx	ZDGLg+8bfUe3RUhfZBZEVA==	Alex Johnson
mbrooks@gmail.org	Lake John	346.229.5141	xxxxxx	oU59az+E9Nh10ZPkCgKkCg==	Cory Hall
shelia43@gmail.net	North Jennifer	772.962.8097	xxxxxx	GQOW8ajP5Vz2+tIMmVEWmQ...	Drew Alexander
psoto@gmail.org	Lake Carmenville	001-507-812-5297x276	xxxxxx	cXVrpOZ14Ee2ktmedGksPQ==	Eric Kirk
sjohnson@gmail.com	Connorview	001-616-713-4005x954	xxxxxx	9VpfuRvwt4EofdTSjaaTUQ==	Eric Guzman
enichols@gmail.com	Robinchester	(380)594-9493	xxxxxx	jI/S1g3HaUU7Njt42vxAfA==	John Tyler
howardtran@gmail.org	Lake Danielburgh	+1-208-865-3818x1697	xxxxxx	IE9AoBbyMj8Zq959fPnFNg==	Kyle Moran

In data fusion services I have transformed the data, here we can validate it.

1. We replace the hostname to gmail.com from example.com which python faker library have created
2. We masked the salary column
3. We used hashing in password column to hide the real password.
4. At last, we combined first and last name to the full name

Data Visualization using Looker Studio



BigQuery

By Google

BigQuery is Google's fully managed, petabyte scale, low-cost analytics data warehouse. BigQuery charges for querying/processing data. Those queries are charged to the credit card of the billing project.

[LEARN MORE](#)

[REPORT AN ISSUE](#)

RECENT PROJECTS	Project	Data set	Table
MY PROJECTS	Enter Project ID manually	employee	employee-data
SHARED PROJECTS	data-pipeline		
CUSTOM QUERY	airflow-project		
	My First Project		
PUBLIC DATA-SETS	My First Project		

After validating the transformation we imported data into looker studio for visualization.

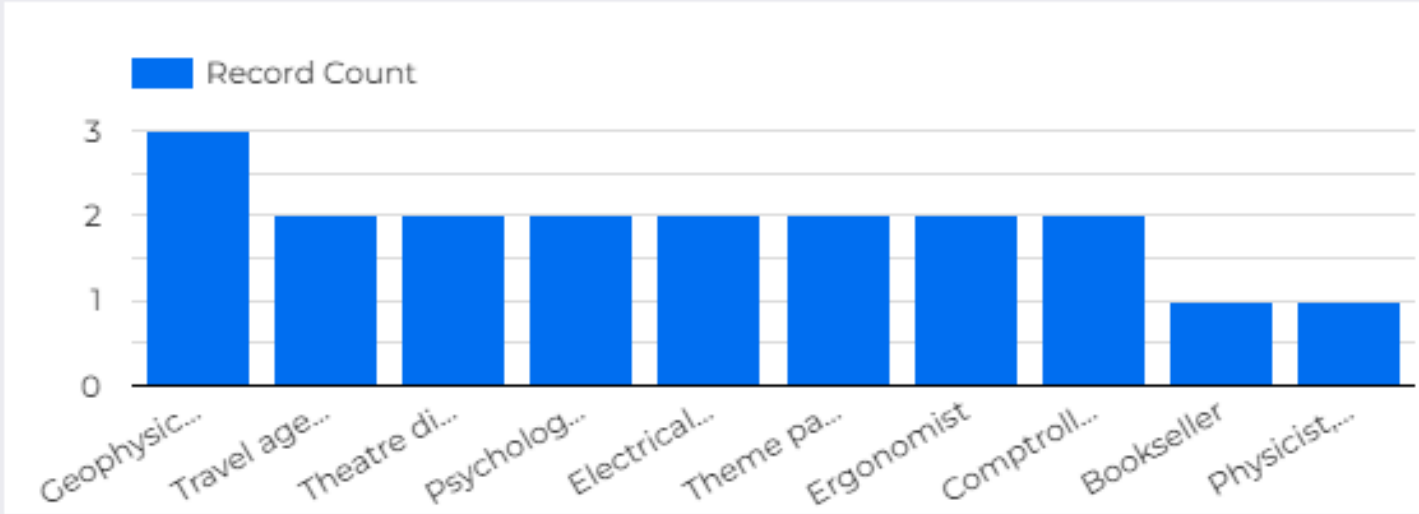
Visualization

Employee Details Summary

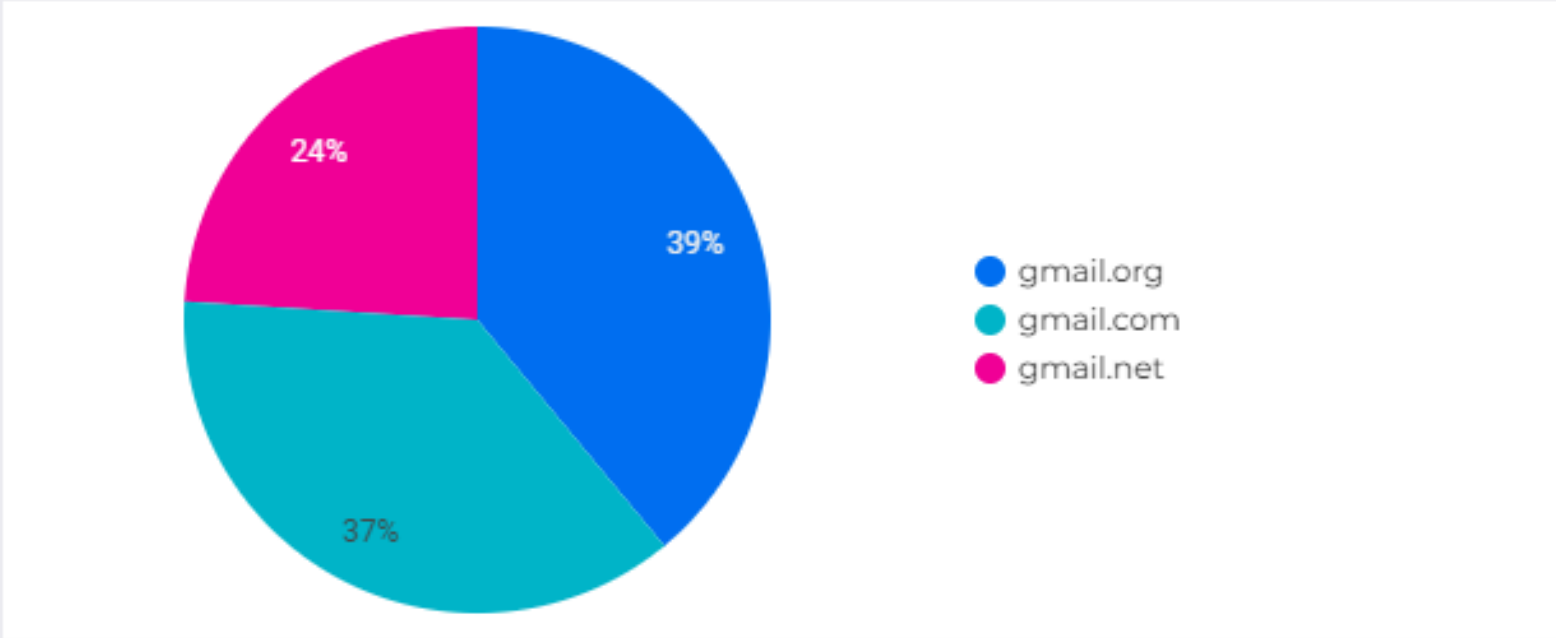
Employee Details

	full_nam...	department	job_title	email
1.	Zachary K...	Horticultural consultant	Surveyor, quantity	trevor26@gmail.com
2.	Wendy Oc...	Journalist, broadcasting	Civil engineer, co...	uedwards@gmail.c...
3.	Wayne Ev...	Administrator, Civil Service	Therapist, drama	jjohnson@gmail.org
4.	Wanda H...	Associate Professor	Sales professional...	phillipbest@gmail.n...
5.	Taylor Bro...	Hospital doctor	Insurance risk sur...	perrybarbara@gmai...
1 - 100 / 100 < >				

Job Title and Department Distribution



Email Domain Analysis



Geographic Distribution



Cloud Composer and Dag

Google Cloud

data-pipeline

Search (/) for resources, docs, products, and more

Composer

Environments

+

CREATE

REFRESH

DELETE

Filter

Filter environments

<div><div></div></div> State	Name <div></div>	Location	Composer version	Airflow version	Creation time	Update time	Airflow webserver	DAG list	Logs	DAGs folder
<div><div></div><div></div></div>	<div><div></div><div>composer-dev</div></div>	us-central1	3	2.9.3-build.0	9/25/24, 2:52 PM	9/25/24, 3:29 PM	<div><div></div><div>Airflow</div><div></div></div>	<div><div></div><div>DAGs</div></div>	<div><div></div><div>Logs</div></div>	<div><div></div><div>DAGs</div></div>

Google Cloud Composer is a fully managed workflow orchestration service built on Apache Airflow, designed to automate and manage complex data pipelines. Within Composer, a DAG (Directed Acyclic Graph) defines the structure and sequence of tasks that constitute a data pipeline. The Composer DAG orchestrates the execution of various tasks, such as data extraction, transformation, loading, and triggering other services like Google Cloud Data Fusion pipelines. By specifying dependencies and scheduling within the DAG, Composer ensures that tasks run in the correct order and at the appropriate times. This automation allows for reliable, repeatable, and scalable data workflows, minimizing manual intervention and reducing the potential for errors.

Composer DAGs are responsible for pipeline automation because they provide a programmable and flexible framework to manage task dependencies, handle retries and failures, monitor workflow execution, and integrate seamlessly with other cloud services, thereby streamlining the entire data processing lifecycle.

Dag Script

```
1  from datetime import datetime, timedelta
2  from airflow import DAG
3  from airflow.operators.bash import BashOperator
4  from airflow.providers.google.cloud.operators.datafusion import CloudDataFusionStartPipelineOperator
5  from airflow.utils.dates import days_ago
6
7  default_args = {
8      'owner': 'airflow',
9      'start_date': days_ago(1), # Use a past date or `days_ago`
10     'depends_on_past': False,
11     'email': ['bidhanpanta123@gmail.com'],
12     'email_on_failure': False,
13     'email_on_retry': False,
14     'retries': 1,
15     'retry_delay': timedelta(minutes=5),
16 }
17
18 dag = DAG('employee_data',
19           default_args=default_args,
20           description='Runs an external Python script and triggers a DataFusion pipeline',
21           schedule_interval='@daily',
22           catchup=False)
23
24 with dag:
25     run_script_task = BashOperator(
26         task_id='data_extraction',
27         bash_command='python /home/airflow/gcs/dags/script/data_extraction.py',
28     )
29
30     start_pipeline = CloudDataFusionStartPipelineOperator(
31         location="europe-west2",
32         pipeline_name="employee-pipeline",
33         instance_name="data-fusion-dev",
34         task_id="start_pipeline",
35     )
36
37     run_script_task >> start_pipeline
38
```

Created DAG script:

Importing Libraries:

- The script imports necessary libraries from datetime, airflow, and airflow.operators to define the DAG and handle tasks.
- The BashOperator is used to run external scripts, and CloudDataFusionStartPipelineOperator is used to trigger Google Cloud Data Fusion pipelines.

Default Arguments (default_args):

- Defines standard configurations for the DAG, such as the owner (airflow), the start date (set to a day ago), email notifications (disabled), retry attempts (1 retry with a 5-minute delay), and preventing the DAG from depending on past runs.

DAG Creation:

- The DAG is named 'employee_data' and is scheduled to run daily (@daily) without backfilling past runs (catchup=False).

Task 1: Running an External Python Script:

- The first task, run_script_task, uses the BashOperator to execute a Python script (data_extraction.py) located in the DAGs folder. This script presumably handles the employee data extraction process.

Task 2: Starting a Data Fusion Pipeline:

- The second task, start_pipeline, uses CloudDataFusionStartPipelineOperator to trigger a Google Cloud Data Fusion pipeline named "employee-pipeline". The pipeline runs in the region europe-west2 within the Data Fusion instance "data-fusion-dev".

Task Dependency:

- The script ensures that the Python data extraction script runs first, and upon its successful completion, the Data Fusion pipeline is triggered (run_script_task >> start_pipeline).

Cloud Composer and Dag

europa-west2-emp-datacompos-94310879-bucket

Location

Storage class

Public access

Protection

europa-west2 (London)

Standard

Subject to object ACLs

Soft Delete

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

INVENTORY REPORTS

OPERATIONS

Folder browser

europa-west2-emp-datacompos-94310879-bucket

dags/

script/

data/

logs/

plugins/

Buckets > europa-west2-emp-datacompos-94310879-bucket > dags

CREATE FOLDER

UPLOAD

TRANSFER DATA

OTHER SERVICES

Filter by name prefix only

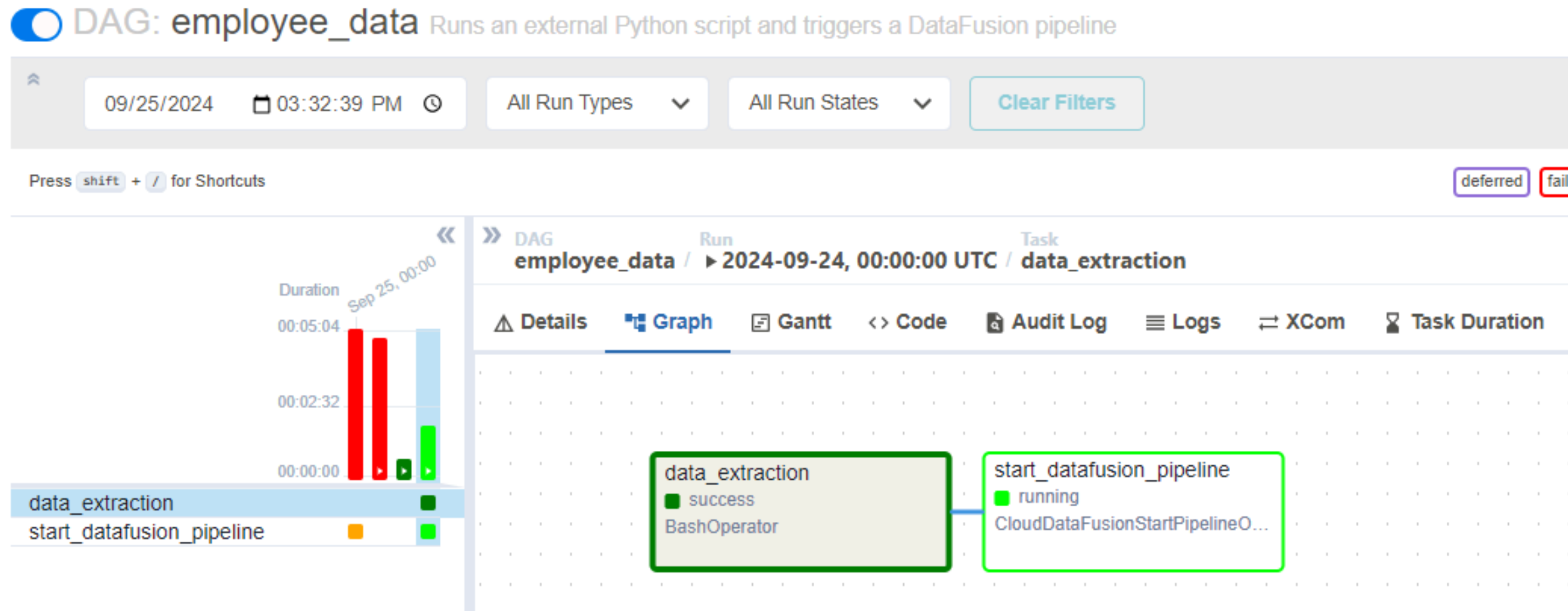
Filter

Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created ?
<input type="checkbox"/>	<div><div></div>airflow_monitoring.py</div>	809 B	text/x-python	Sep 25, 2024, 6:01:46 PM
<input type="checkbox"/>	<div><div></div>dag.py</div>	1.2 KB	application/octet-stream	Sep 25, 2024, 6:22:32 PM
<input type="checkbox"/>	<div><div></div>script/</div>	—	Folder	—

After creating DAG we upload it to composer dag folder **'dag.py'** and we upload employee python code inside the script folder then it creates the automation pipeline inside the apache airflow **'employee_data'** where we can monitor our pipeline based on scheduled time in our case it is triggered daily.

Apache Airflow



Apache Airflow is an open-source platform used to create, schedule, and monitor complex workflows or data pipelines. It enables users to define workflows as code using Directed Acyclic Graphs (DAGs), which specify the sequence and dependencies of tasks. It also offers built-in features for scheduling, error handling, logging, and monitoring through an intuitive web interface. By automating data processing, machine learning, and other repetitive tasks, Airflow helps streamline operations and ensures scalable, reliable workflow management. In above figure we can see successfully transmission of data to the bucket and then to the bigquery and then to the looker studio through pipeline automation.



Thank You

By: Bidhan Pant