



**Manage Data
Pipelines with Cloud
Data Fusion and Cloud
Composer**

Manage Data Pipelines with Cloud Data Fusion and Cloud Composer

01

- Building batch data pipelines visually with Cloud Data Fusion
- Components
 - UI overview
 - Building a pipeline
 - Exploring data using Wrangler

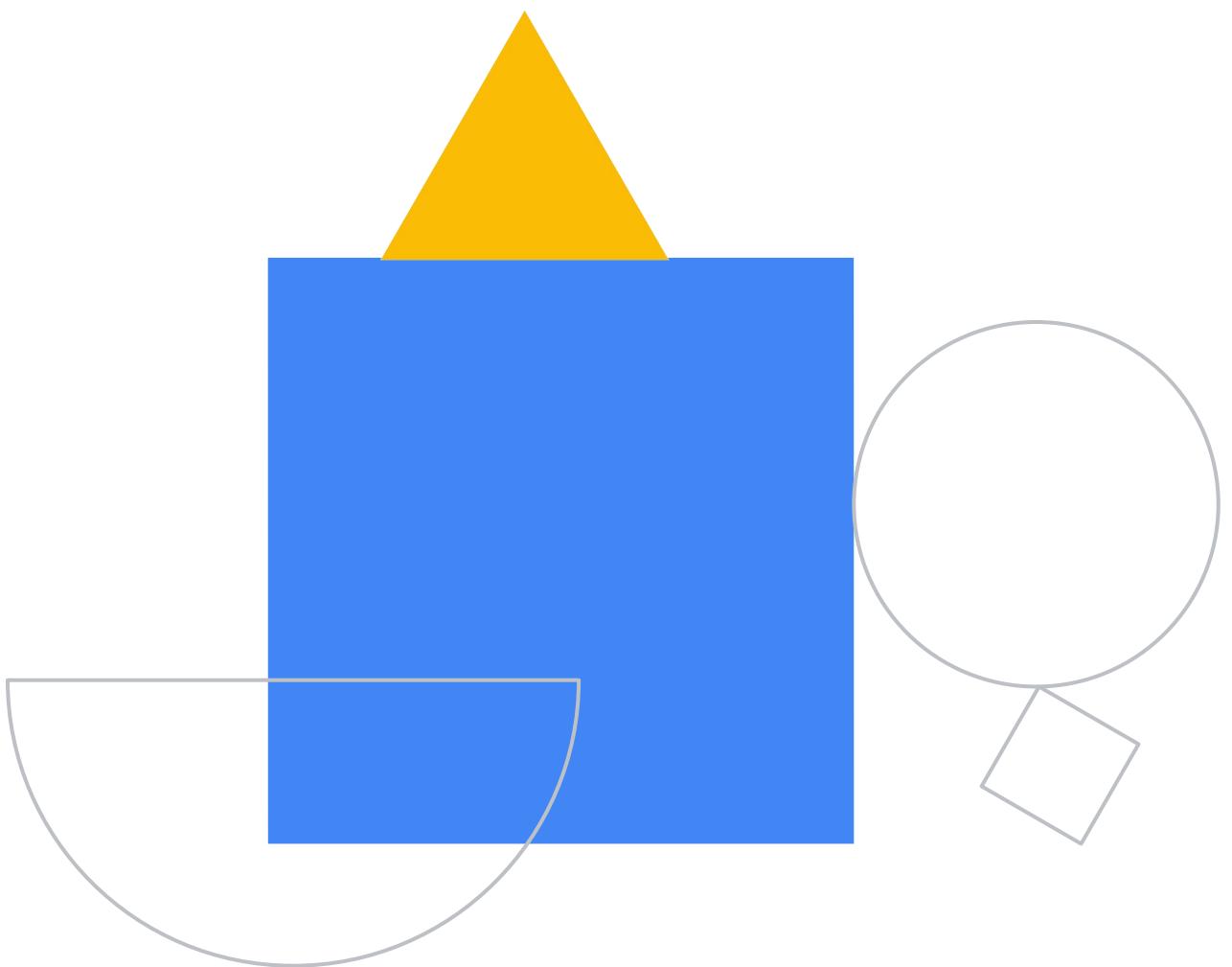
02

- Orchestrating work between Google Cloud services with Cloud Composer
- Apache Airflow environment
 - DAGs and Operators
 - Workflow Scheduling
 - Monitoring and Logging



Lab Startup

Building and Executing a Pipeline Graph in
Cloud Data Fusion



Manage Data Pipelines with Cloud Data Fusion and Cloud Composer

01

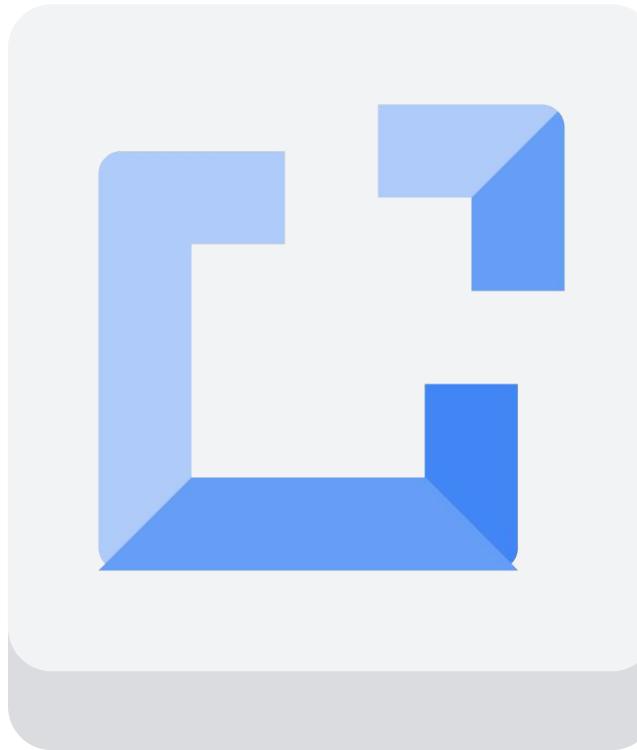
- Building batch data pipelines visually with Cloud Data Fusion
 - Components
 - UI overview
 - Building a pipeline
 - Exploring data using Wrangler

02

- Orchestrating work between Google Cloud services with Cloud Composer
 - Apache Airflow environment
 - DAGs and Operators
 - Workflow Scheduling
 - Monitoring and Logging



Cloud Data Fusion



Cloud Data Fusion is a **fully-managed, cloud native, enterprise data integration service** for quickly building and managing data pipelines.

Developer, data scientist, and business analyst



Need to cleanse, match, de-dupe, blend, transform, partition, transfer, standardize, automate, and monitor.



Use Cloud Data Fusion to visually build integration pipeline, test, debug and deploy.



Run it at scale on Google Cloud, operationalize (monitor, report) pipelines, inspect rich integration metadata.

Benefits



Integrate with any data



Increase productivity



Reduce complexity

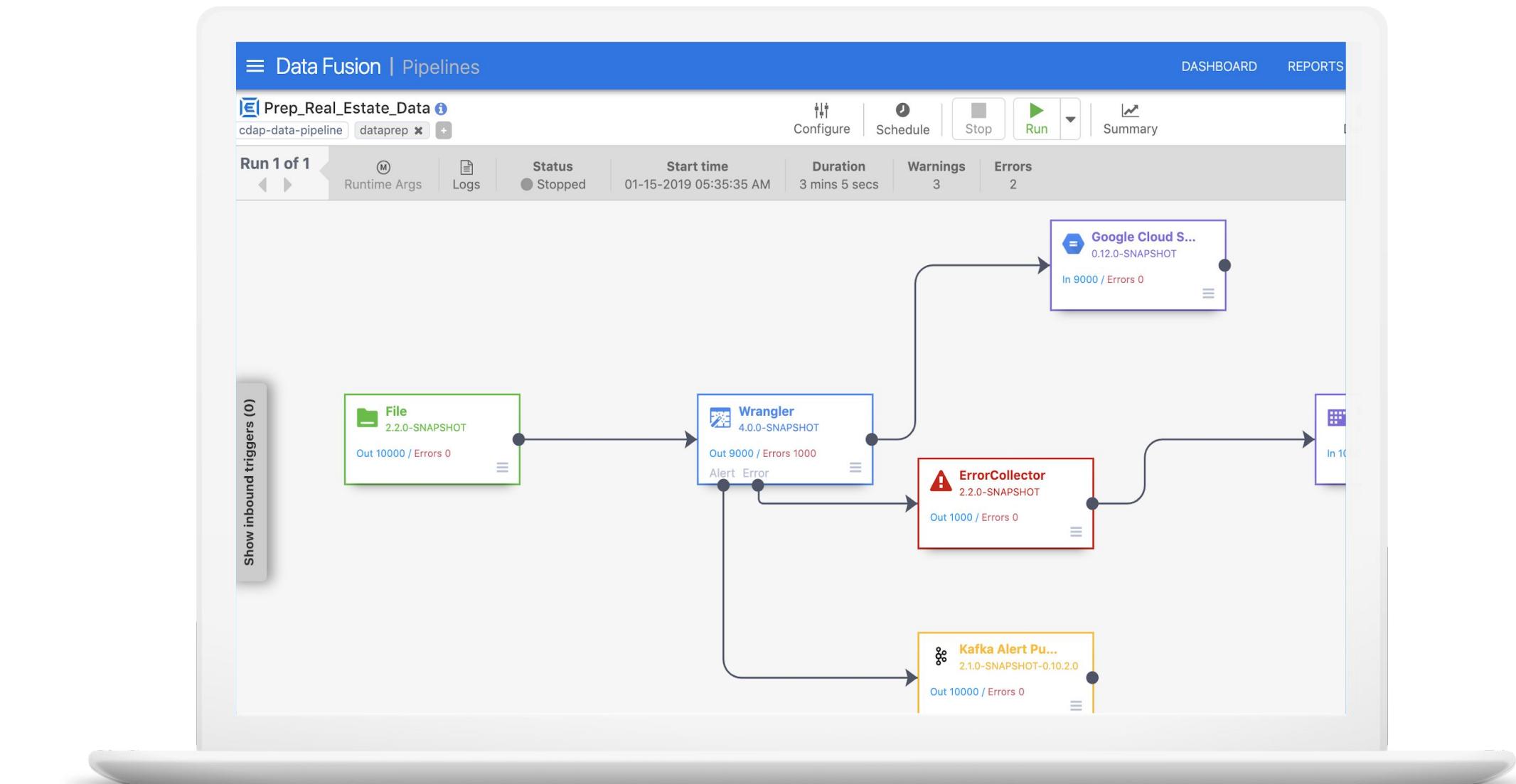


Increase flexibility

	Double	String	String	String	String	Double	Integer	Long	Integer
	▼ price □	▼ city □	▼ zip □	▼ type □	▼ beds □	▼ baths □	▼ size □	▼ lot_size □	▼ stories □
1	1000000.0	Santa Clara	95050	Condo	2	2.5	1410	1422	3
2	1000000.0	Santa Clara	95050	Condo	3	2.5	1670	1740	2
3	1000000.0	Santa Clara	95050	Condo	3	2.5	1708	1750	2
4	1000000.0	Santa Clara	95051	Single-Family Home	3	2.0	1068	5600	1
5	1000000.0	Palo Alto	94306	Condo	2	1.5	998	499	2
6	1000000.0	Sunnyvale	94089	Single-Family Home	3	2.0	1108	5824	1
7	1000000.0	Santa Clara	95054	Single-Family Home	3	2.0	1612	6250	1
8	1000000.0	Mountain View	94040	Condo	2	2.0	1206	1880	1
9	1000000.0	Sunnyvale	94085	Condo	2	2.5	1198	1082	3
10	1000000.0	Sunnyvale	94085	Condo	3	3.5	1513	1575	3
11	1000000.0	Santa Clara	95054	Single-Family Home	3	2.0	1097	6200	1

Build data pipelines with a friendly UI

- Rich graphical interface
- **100+ plugins** - connectors, transforms, and actions
- **Code free** visual transformations
- **1000+ transforms**, data quality
- Test and debug pipeline
- Pre-built pipelines
- Developer SDK



Integration metadata

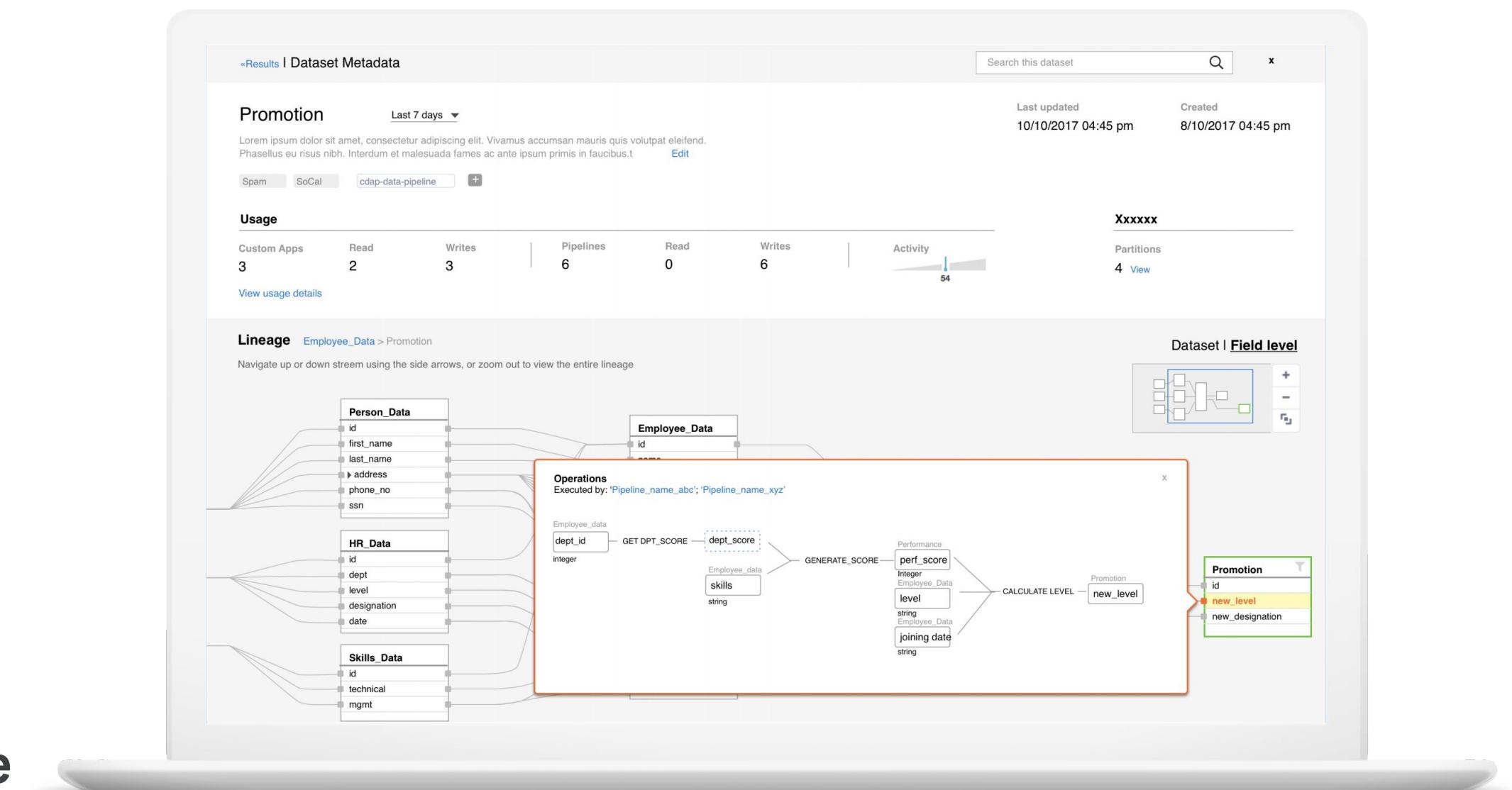
Tags and Properties support

- Pipeline
- Dataset
- Schema

Search integrated entities by

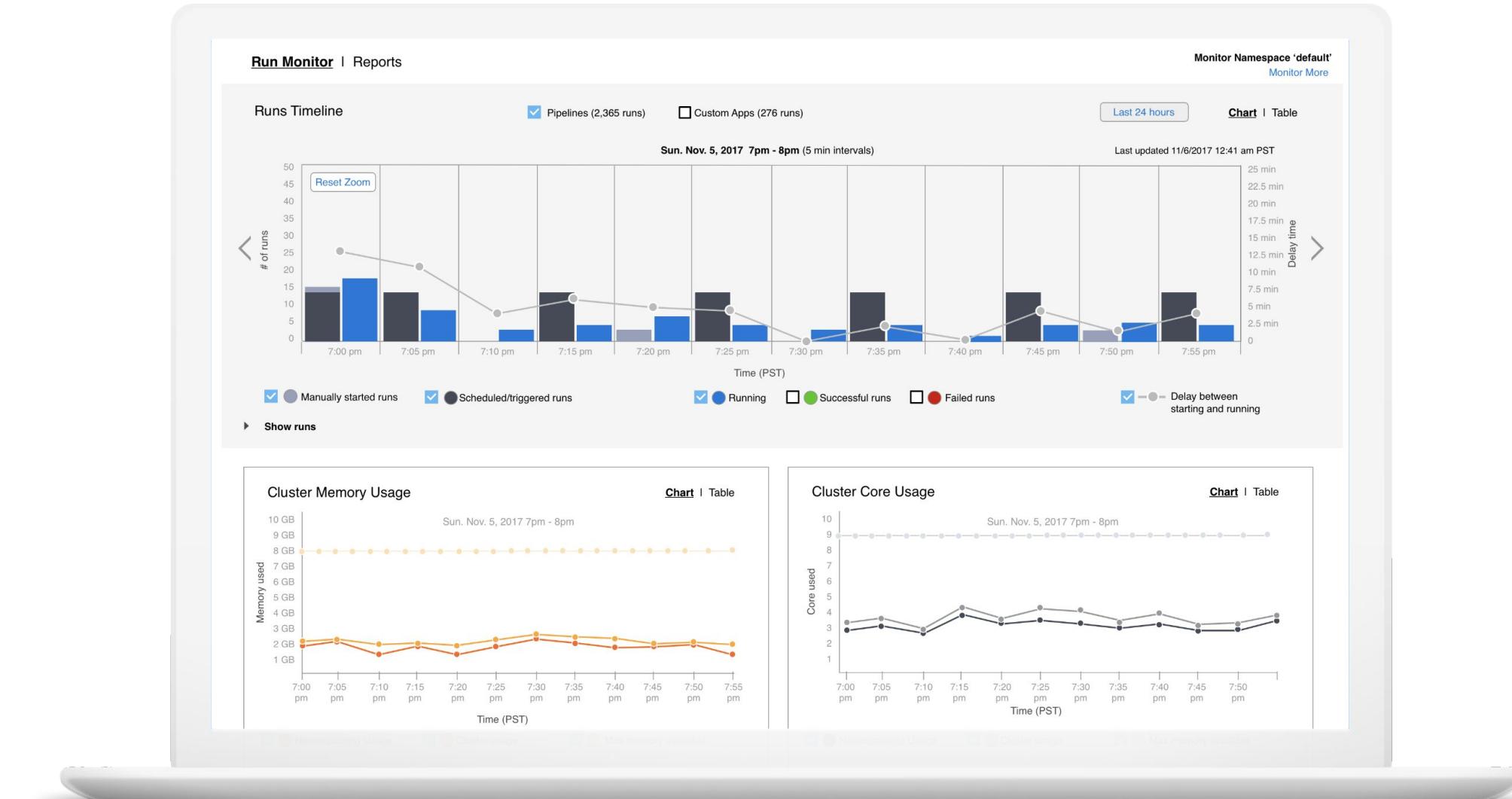
- Keyword
- Schema name and type

Dataset level and Field level lineage



Extensible

- Pipeline templatization
- Conditional pipeline triggers
- Plugin management
- Plugin templatization
- Plugin UI widget
- Custom provisioners
- Custom compute profiles
- Hub integration



Components of Cloud Data Fusion

CASK Overview Data Preparation Pipelines Metadata

Create Pipeline Ingest Data View Schema

titanic.csv

	String	String	String	String	String	String	String	String	String	String	String
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Cabin	Fare
1	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171		
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599		
3	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 31		
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803		
5	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450		
6	6	0	3	Moran, Mr. James	male		0	0	330877		
7	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463		
8	8	0	3	Palsson, Master, Gosta Leonard	male	2	3	1	349909		
9	9	1	3	Johnson, Mrs. Oscar W (Eliabeth Vilhelmina Berg)	female	27	0	2	347742		
10	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736		
11	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549		
12	12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783		
13	13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151		
14	14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082		
15	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406		
16	16	1	2	Hawley, Mrs. (Mano D Kinchima)	female	55	0	0	248706		

COPYRIGHT © 2017 CASK DATA, INC.

01

Wrangler – Framework

- Data Preparation for on-boarding new sources and datasets.
- Perform Data Transformations, Data Quality checks with visual feedback.
- Extend the Wrangler by building new user defined directives.
- Integrates with Data Pipeline for operationalizing transformations.

02

Data Pipeline – Framework

-
- SSGCompositeProcessor v2
Ingest BBG, IDC and Reuters files into HBase Tables and also to HDFS Files to create SSG Composite using Rules Engine.
- Run 1 of 1 • Succeeded Duration 17 secs Start Time 08:21:00 pm
- SSG Driver 1.0.0-SNAPSHOT
Records Out: 0 / Errors: 0
Error Alert
- BBG 1.7.0
Records Out: 12 / Errors: 3
Error Alert
- IDC 1.7.0
Records Out: 10 / Errors: 3
Error Alert
- Reuters 1.7.0
Records Out: 10 / Errors: 3
Error Alert
- BBG Transform 2.0.0
Records Out: 11 / Errors: 0
Error Alert
- IDC Transform 2.0.0
Records Out: 9 / Errors: 3
Error Alert
- TR Transform 2.0.0
Records Out: 9 / Errors: 3
Error Alert
- Joiner 1.7.0
Records Out: 7 / Errors: 0
Error Alert
- RulesEngine 1.0.0-SNAPSHOT
Records Out: 7 / Errors: 0
Error Alert
- Composite Table 1.7.0
Records Out: 7 / Errors: 0
Error Alert
- COPYRIGHT © 2017 CASK DATA, INC.
- User interface for building complex data workflows.
 - Join, Lookup, Aggregate, Filtering data in-flight.
 - Building complex workflows with 100s of connectors.
 - Extend Data Pipeline using simple APIs.
 - Integrates with Dataprep, Rule Engine and Metadata Aggregator.

Components of Cloud Data Fusion

The screenshot shows the Cask Rules Engine interface. At the top, there are tabs for Overview, Data Preparation, Pipelines, and Metadata. Below that, there are two sections: 'Rules Books (5)' and 'Rules (23)'. The 'Rules Books' section contains a search bar and a 'Create a New Rule' button. The 'Rules (23)' section shows a table with columns for Name, Last Updated, and Description. One row is selected, showing a detailed view of a rule named 'asset-id'. The detailed view includes the rule's description: 'This defines the rulebook for the client composite dataset generation for SSG'. Below this, there is a table titled 'Rules (9)' listing nine specific rules with their descriptions and removal options.

The screenshot shows the Cask Metadata Aggregator interface. At the top, there are tabs for Overview, Data Preparation, Pipelines, and Metadata. Below that, there are sections for rulebook, summary, audit log, preview, usage, and compliance. A green '+' button is visible. The main area displays a lineage diagram for a dataset named 'rulebook'. The diagram shows a 'yare service' node connected to two 'Table' nodes: 'rulebook' and 'rules'. Arrows indicate the flow from the service to each table.

03

Rules Engine — Tool

- Business Data Transformations and checks codified for business users.
- Define Complex rules using intuitive and simple to use user interface.
- Logically group Rules in Rulebook and trigger or schedule processing.
- Integrates with Data Pipeline for operationalizing Rules.

04

Metadata Aggregator — Tool

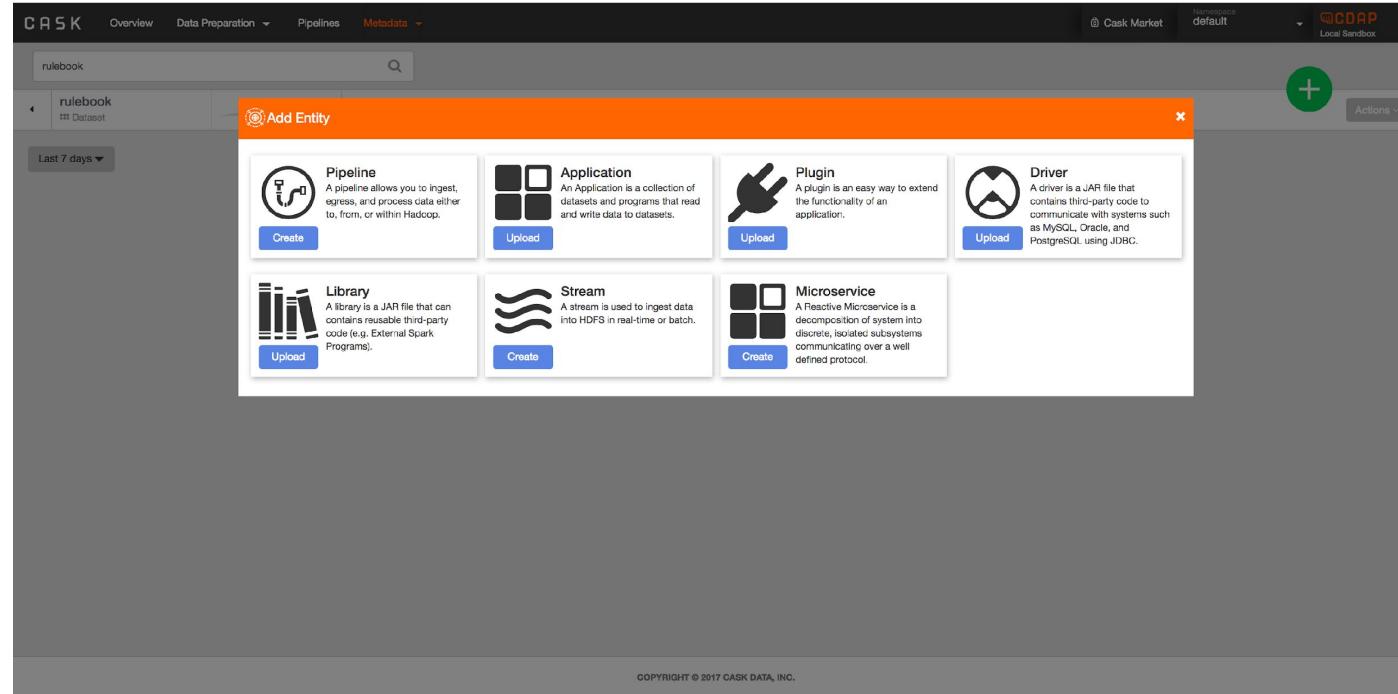
- Aggregate Business, Technical and Operational Metadata.
- Track the flow of data (Lineage) for richer data needed for governance.
- Create Data Dictionary and Metadata Repository.
- Integrate with enterprise MDM solutions.
- Integrates with Data Pipeline, Rules Engine.

Components of Cloud Data Fusion

05

Microservice — Framework

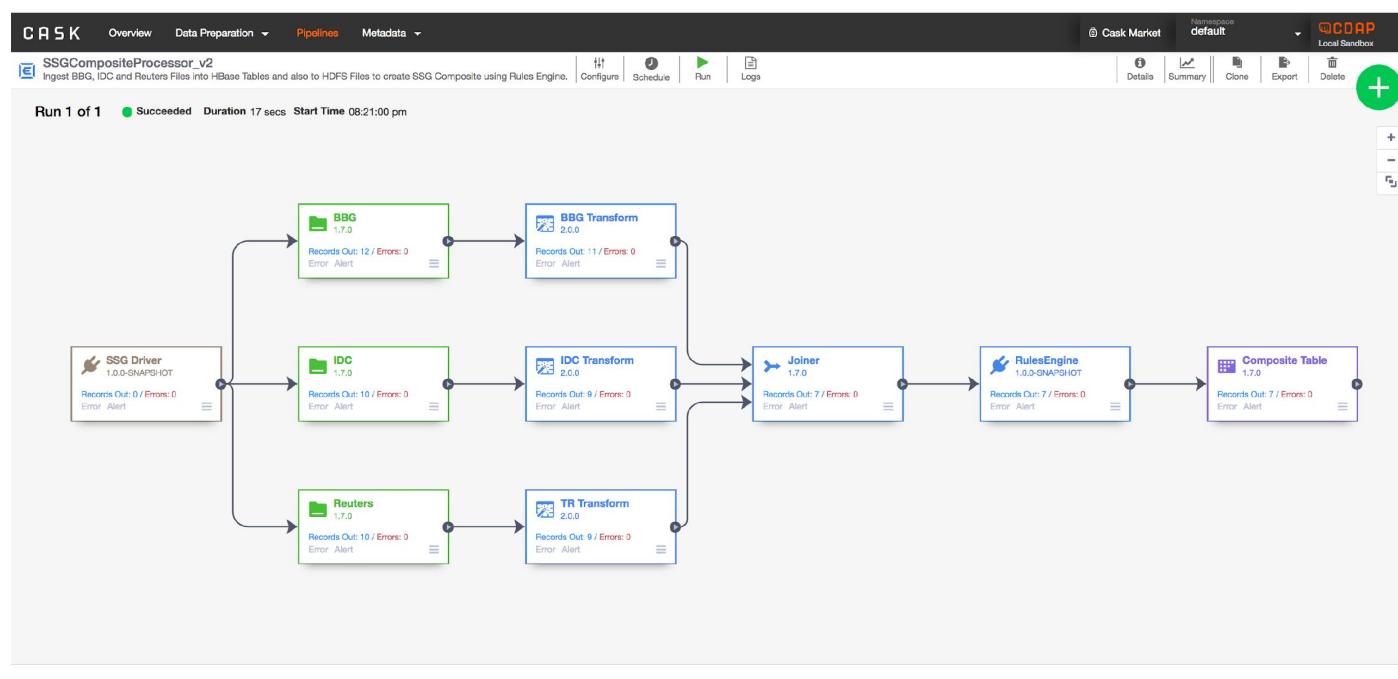
- Build specialized logic for processing data.
- Create loosely coupled network for processing events.
- Bind processing to varied set of queues.



06

Event Condition Action (ECA) — Application

- Delivers a specialized solutions for IoT event processing.
- Parses any events, triggers conditions and executes Action.
- Real-time notification system, with easy-to-use user interface for configuring event parsing, condition and actions.



Cloud Data Fusion - UI overview

- Control Center
- Pipelines
- Wrangler
- Metadata
- Hub
- Entities
- Administration

Type	Status	Last start time	Next run in	Total runs	Tags
Batch	Failed	10-18-2018 05:01:35 PM	58 sec	2	Wrangler Real_Estate cdap-data-pipeline
Batch	Running	10-18-2018 05:01:35 PM	1 hr	10	Wrangler Real_Estate cdap-data-pipeline
Batch	Succeeded	10-18-2018 05:01:35 PM	1 day	13	cdap Real_Estate SoCal cdap-data-pipeline
Realtime	Deployed	--	--	--	cdap-data-pipeline
Realtime	Running	10-18-2018 05:01:35 PM	--	234	Wrangler Real_Estate new Spam SoCal cdap-data-pipeline
Batch	Failed	10-18-2018 05:01:35 PM	--	35	cdap-data-pipeline
Batch	Succeeded	10-18-2018 05:01:35 PM	1 hr	5,678	cdap-data-pipeline
Realtime	Succeeded	10-18-2018 05:01:35 PM	1 month	345	cdap-data-pipeline
Batch	Succeeded	10-18-2018 05:01:35 PM	--	1	cdap-data-pipeline
Batch	Succeeded	10-18-2018 05:01:35 PM	--	24	cdap-data-pipeline

Control Center

- Application
- Artifact
- Dataset

Cloud Data Fusion | Control Center

DASHBOARD REPORTS HUB SYSTEM ADMIN

Entities in namespace "default"

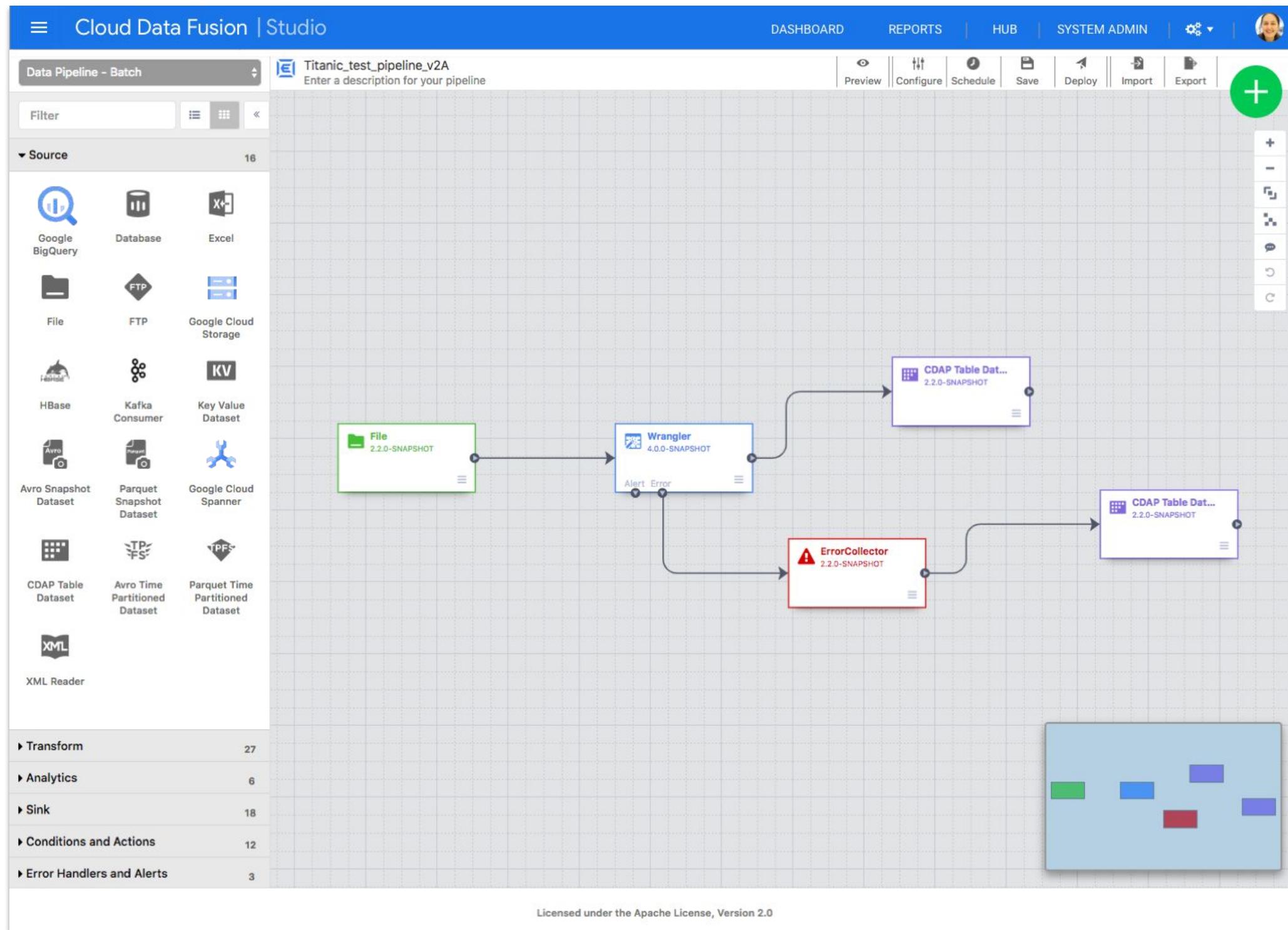
Displaying Applications, Datasets, sorted by Newest

26 entities

Data Pipeline join-w-customer 1.0.0-SNAPSHOT	Dataset sales-with-zip	Dataset customers.csv	Dataset sales.csv	Data Pipeline sales-ingest 1.0.0-SNAPSHOT
Programs: 2 Running: 0 Failed: 0	Programs: 1 Operations: 10 Writes: 3	Programs: 1 Operations: 0 Writes: 0	Programs: 1 Operations: 0 Writes: 0	Programs: 2 Running: 0 Failed: 0
Dataset sales	Data Pipeline customers-ingest 1.0.0-SNAPSHOT	Dataset customers	Dataset U.S._Chronic_Disease_Indicator...	Data Pipeline Mask_data 1.0.0-SNAPSHOT
Programs: 2 Operations: 15 Writes: 10	Programs: 2 Running: 0 Failed: 0	Programs: 2 Operations: 12 Writes: 9	Programs: 1 Operations: 0 Writes: 0	Programs: 2 Running: 0 Failed: 0
Dataset Mask_data	Application ModelManagementApp 1.0.0-SNAPSHOT	Dataset experiment_model_meta	Dataset experiment_model_components	Dataset experiment_meta
Programs: 1 Operations: 42,119,097 Writes: 42,119,097	Programs: 1 Running: 0 Failed: 0	Programs: 1 Operations: 0 Writes: 0	Programs: 1 Operations: 0 Writes: 0	Programs: 1 Operations: 1 Writes: 0
Dataset experiment_splits	Data Pipeline Titanic_test_pipeline_v1_test_v1 1.0.0-SNAPSHOT	Dataset titanic.csv	Data Pipeline Titanic_test_pipeline_v1_test 1.0.0-SNAPSHOT	Dataset Titanic_test
Programs: 1 Operations: 0 Writes: 0	Programs: 2 Running: 0 Failed: 0	Programs: 0 Operations: 0 Writes: 0	Programs: 2 Running: 0 Failed: 0	Programs: 0 Operations: 0 Writes: 0
Dataset Error_sink_titanic	Application dataprep 1.0.0-SNAPSHOT	Dataset connections	Dataset dataprepfs	Dataset workspace
Programs: 0 Operations: 0 Writes: 0	Programs: 1 Running: 1 Failed: 0	Programs: 1 Operations: 0 Writes: 0	Programs: 1 Operations: 0 Writes: 0	Programs: 1 Operations: 2,499 Writes: 239

Pipelines

- Developer Studio
- Preview
- Export
- Schedule
- Connector and function palette
- Navigation



Wrangler

- Connections
- Transforms
- Data Quality
- Insights
- Functions

Cloud Data Fusion | Wrangler

DASHBOARD HUB SYSTEM ADMIN Enterprise Edition

titanic.csv Google Cloud Storage titanic.csv

Create a Pipeline More +

	Integer	String	String	String	Integer	String	String	String	Double	String	String
	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	1	0	3	male	22	1	0	A/5 21171	7.25	none	S
2	2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	3	1	3	female	26	0	0	STON/O2. 3101282	7.925	none	S
4	4	1	1	female	35	1	0	113803	53.1	C123	S
5	5	0	3	male	35	0	0	373450	8.05	none	S
6	7	0	1	male	54	0	0	17463	51.8625	E46	S
7	8	0	3	male	2	3	1	349909	21.075	none	S
8	9	1	3	female	27	0	2	347742	11.1333	none	S
9	10	1	2	female	14	1	0	237736	30.0708	none	C
10	11	1	3	female	4	1	1	PP 9549	16.7	G6	S
11	12	1	1	female	58	0	0	113783	26.55	C103	S
12	13	0	3	male	20	0	0	A/5. 2151	8.05	none	S
13	14	0	3	male	39	1	5	347082	31.275	none	S
14	15	0	3	female	14	0	0	350406	7.8542	none	S
15	16	1	2	female	55	0	0	248706	16.0	none	S
16	17	0	3	male	2	4	1	382652	29.125	none	Q

Columns (16) Transformation steps (20)

Search Column names ▾

Name Completion

1 PassengerId 100%

2 Survived 100%

3 Pclass 100%

4 Sex 100%

5 Age 100%

6 SibSp 100%

7 Parch 100%

8 Ticket 100%

9 Fare 100%

10 Cabin 100%

11 Embarked 100%

12 Last_Name 100%

13 Salutation 100%

14 First_Name 100%

15 Today_Fare 100%

16 id 100%

Integration metadata

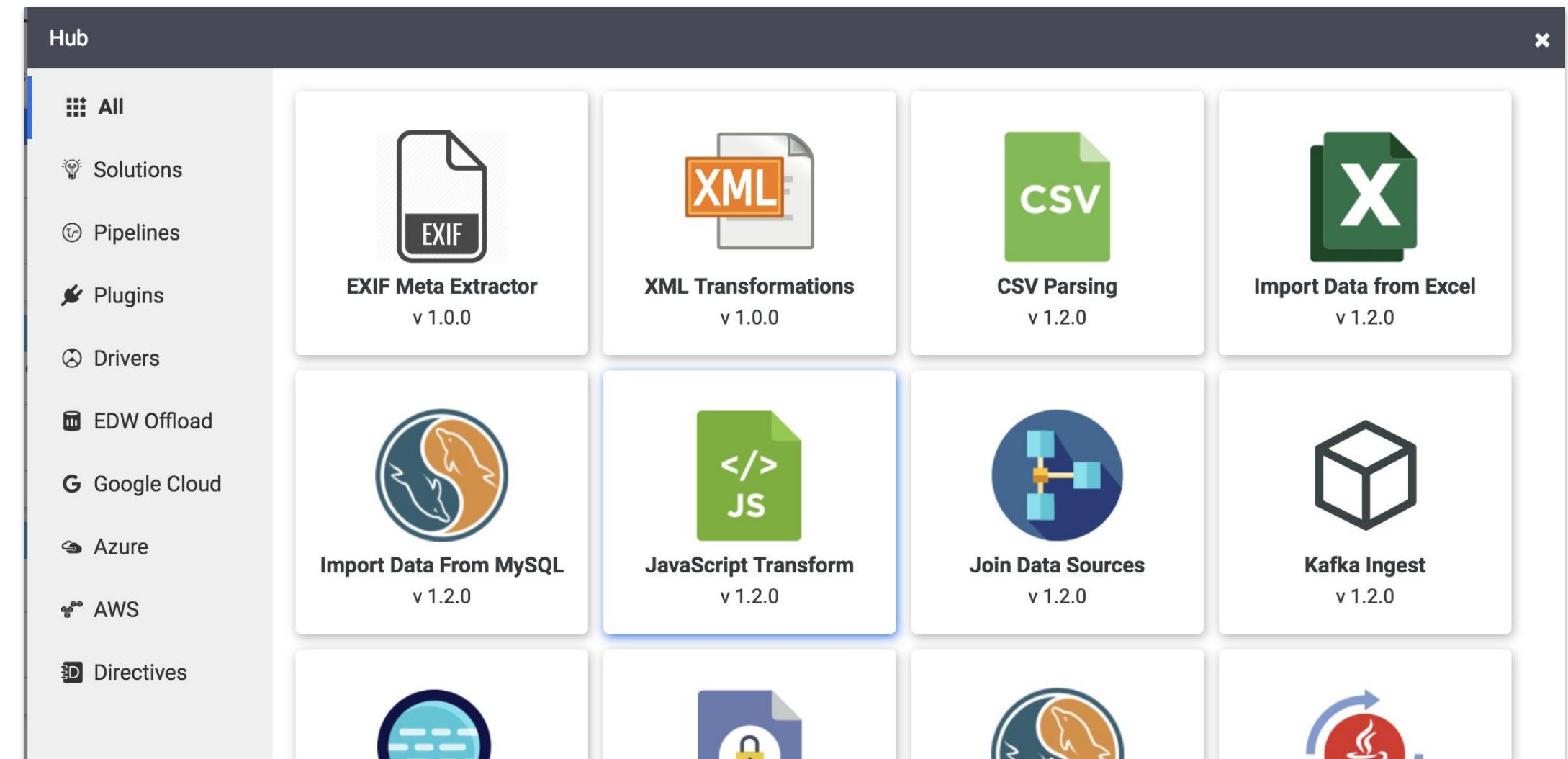
- Search
- Tags and Properties
- Lineage - Field and Data

The screenshot shows the Cloud Data Fusion Metadata interface. The left sidebar displays navigation options: Control Center, Pipelines (with List and Studio), Transform, and Metadata (which is selected). The main area lists 18 datasets, sorted by creation date (Oldest first). The datasets listed are:

- recipes (Dataset, Created: Oct 12, 2018, Recipe store.)
- workspace (Dataset, Created: Oct 12, 2018, Dataprep workspace dataset)
- dataprep (Dataset, Created: Oct 12, 2018, Store Dataprep Index files)
- connections (Dataset, Created: Oct 12, 2018, DataPrep connections store.)
- Error_sink_titanic (Dataset, Created: Oct 17, 2018, No description provided for this Dataset.)
- Titanic_test (Dataset, Created: Oct 17, 2018, No description provided for this Dataset.)
- titanic.csv (Dataset, Created: Oct 17, 2018, No description provided for this Dataset.)

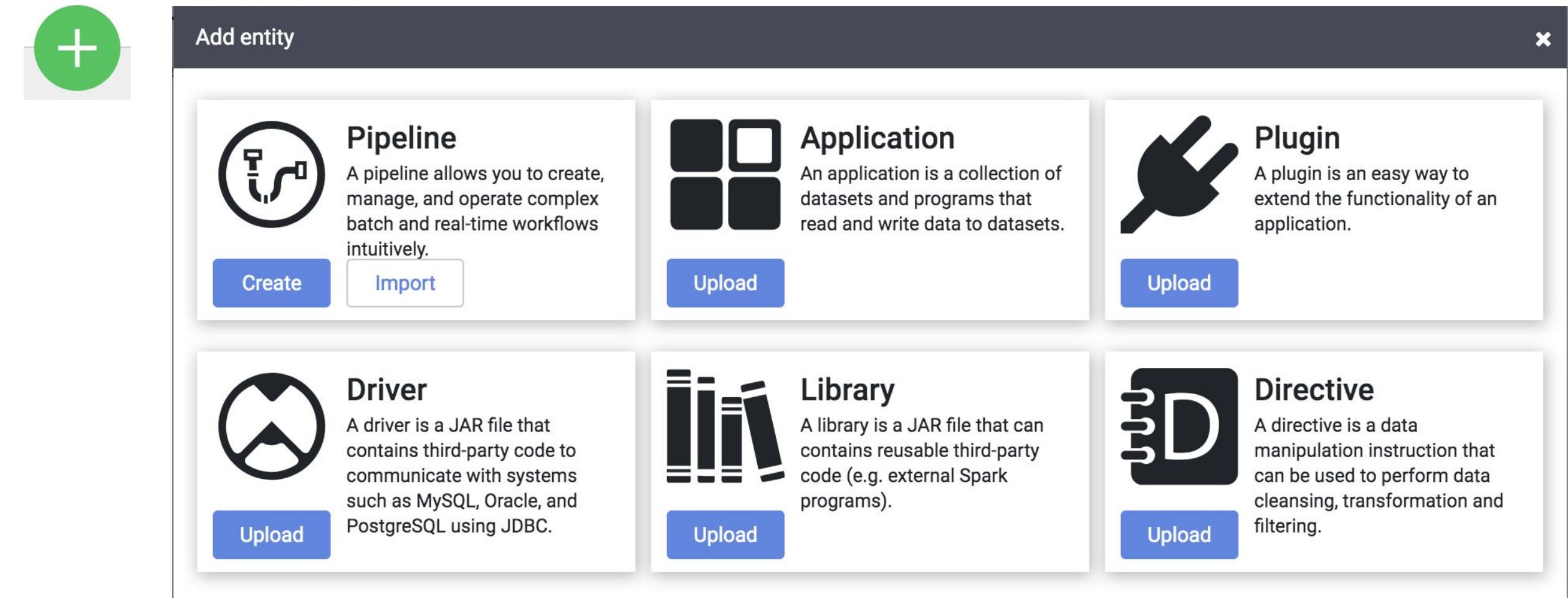
Hub

- Plugins
- Use cases
- Pre-built pipelines



Entities

- Pipeline
- Application
- Plugin
- Driver
- Library
- Directive



Administration

- Management
 - Services
 - Metrics
- Configuration
 - Namespace
 - Compute Profiles
 - Preferences
 - System Artifacts
 - REST Client

Management | Configuration

Services

Status	Name ▾	Provisioned	Requested	Action
Green	App Fabric	1	1	View Logs
Green	Dataset Executor	1	1	View Logs
Green	Log Saver	1	1	View Logs
Green	Messaging Service	1	1	View Logs
Green	Metadata Service	1	1	View Logs
Green	Metrics	1	1	View Logs
Green	Metrics Processor	1	1	View Logs
Green	Transaction	1	1	View Logs

System metrics

Entities	Last hour load
Datasets	0
Programs	0
Namespaces	1
Artifacts	36
Applications	0
ClientErrors	2
ServerError	0
ErrorLogs	2
WarnLogs	35
TotalRequests	192,059
Successful	192,057

Management | Configuration

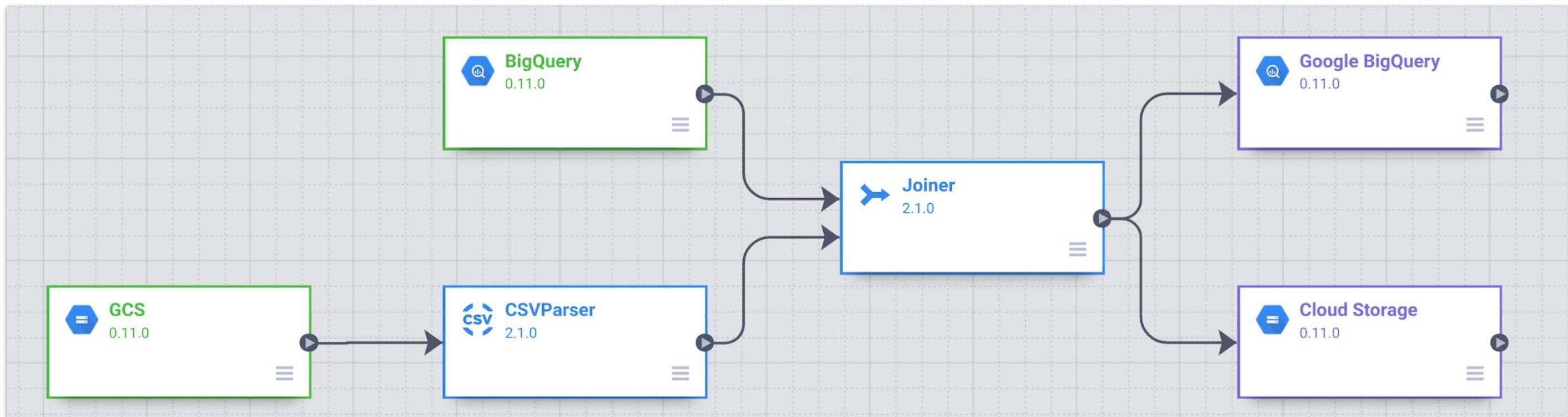
Namespaces (1)

System Compute Profiles (1)

Default		Profile name ▾	Provisioner	Scope	Last 24 hrs runs	Last 24 hrs node hours	Total node hours	Schedules	Triggers	Status
★	Dataproc	Google Cloud Dataproc	SYSTEM	--	--	--	0	0	Enabled	⚙️

System Preferences (1)

Data Pipeline | Directed Acyclic Graph (DAG)

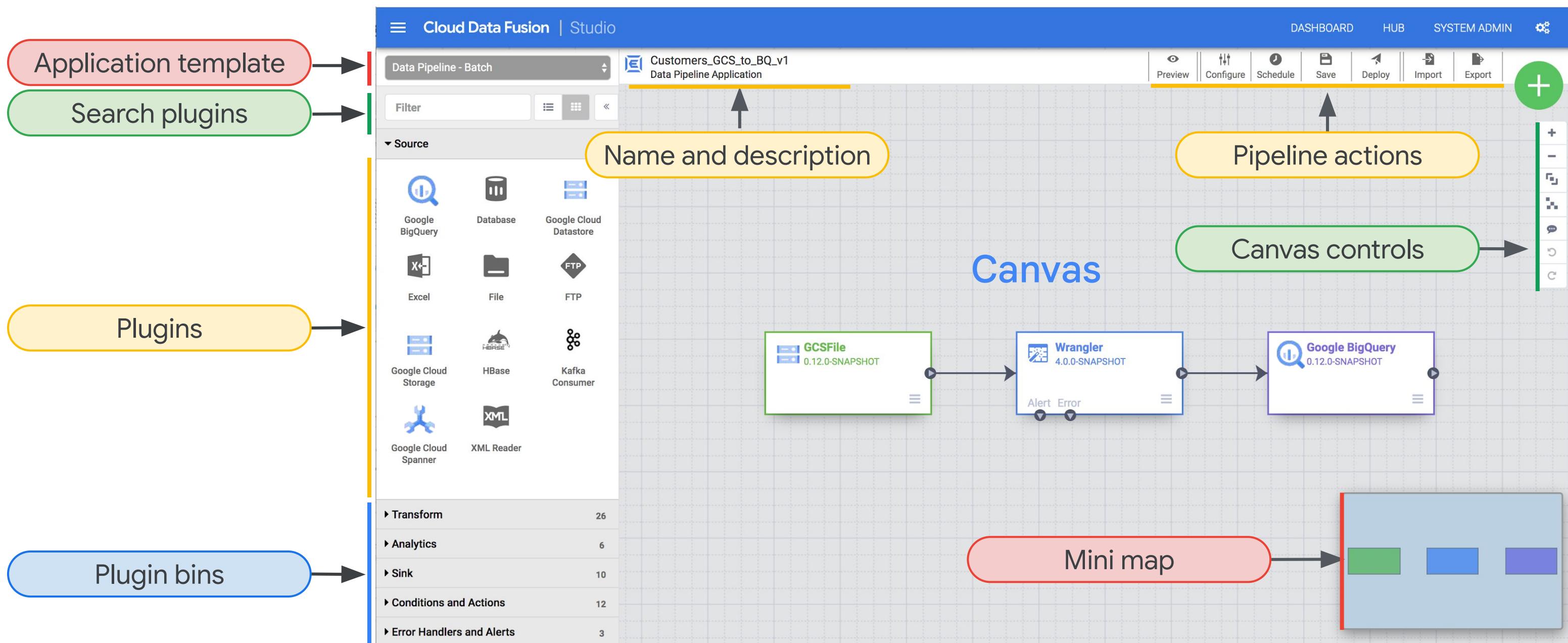


- Represented by a series of stages arranged in a DAG. This forms a one-way pipeline.
- Stages, which are the "nodes" in the pipeline graph, can be of different types.

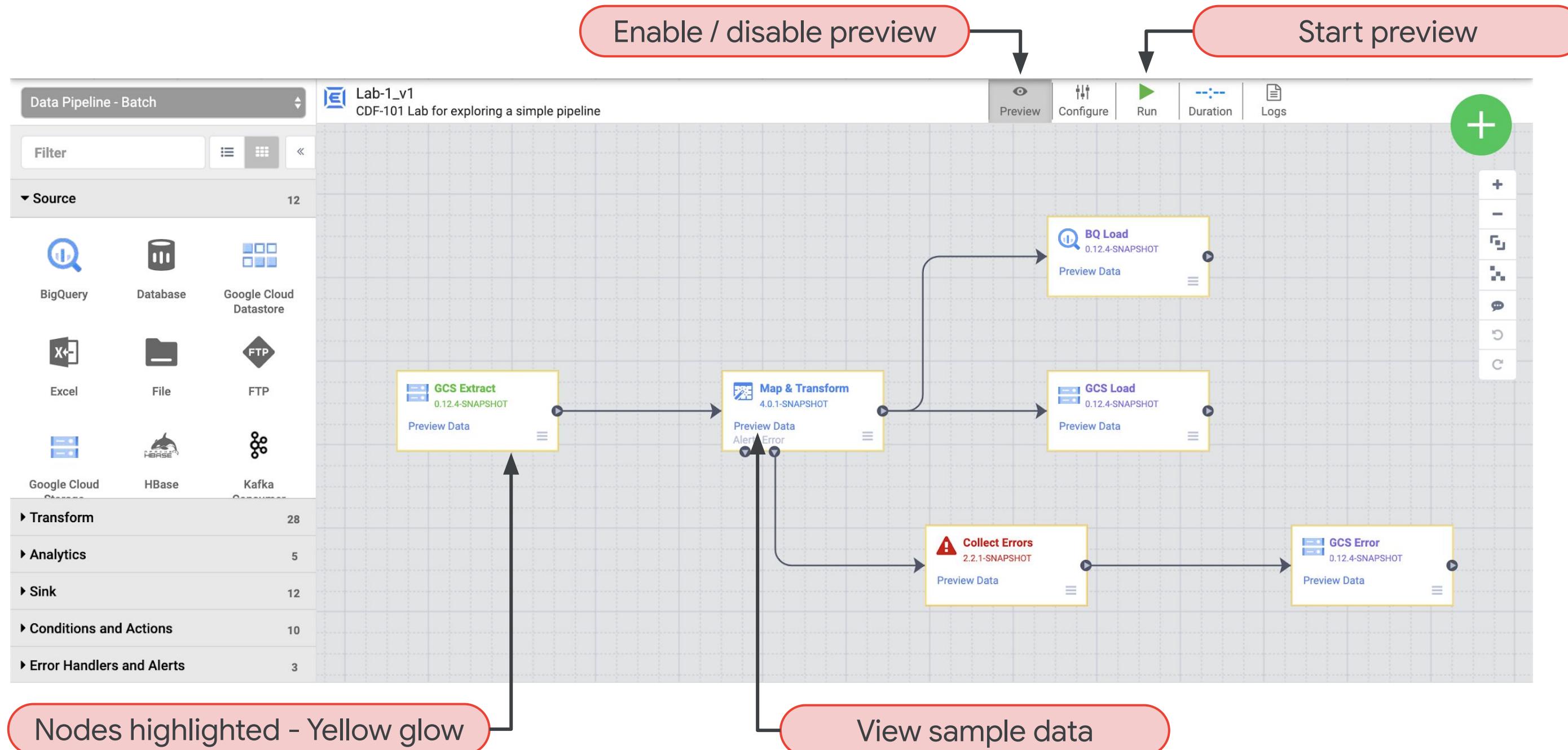
Data pipeline

- Allows non-linear pipelines.
- Can fork from a node, where output from a node can be sent to two or more stages.
- Two or more forked nodes can merge at a transform or a sink node.

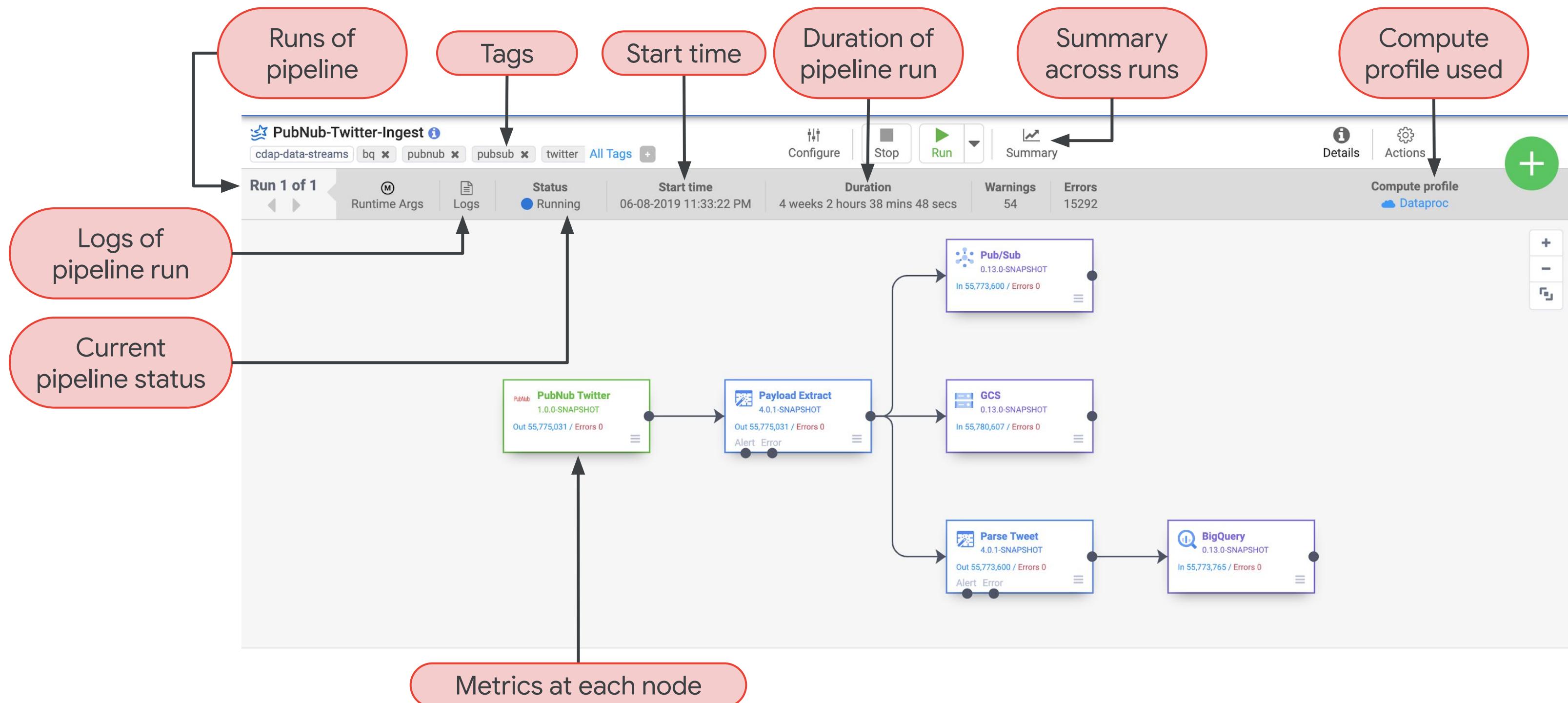
Studio is the UI where you create new pipelines



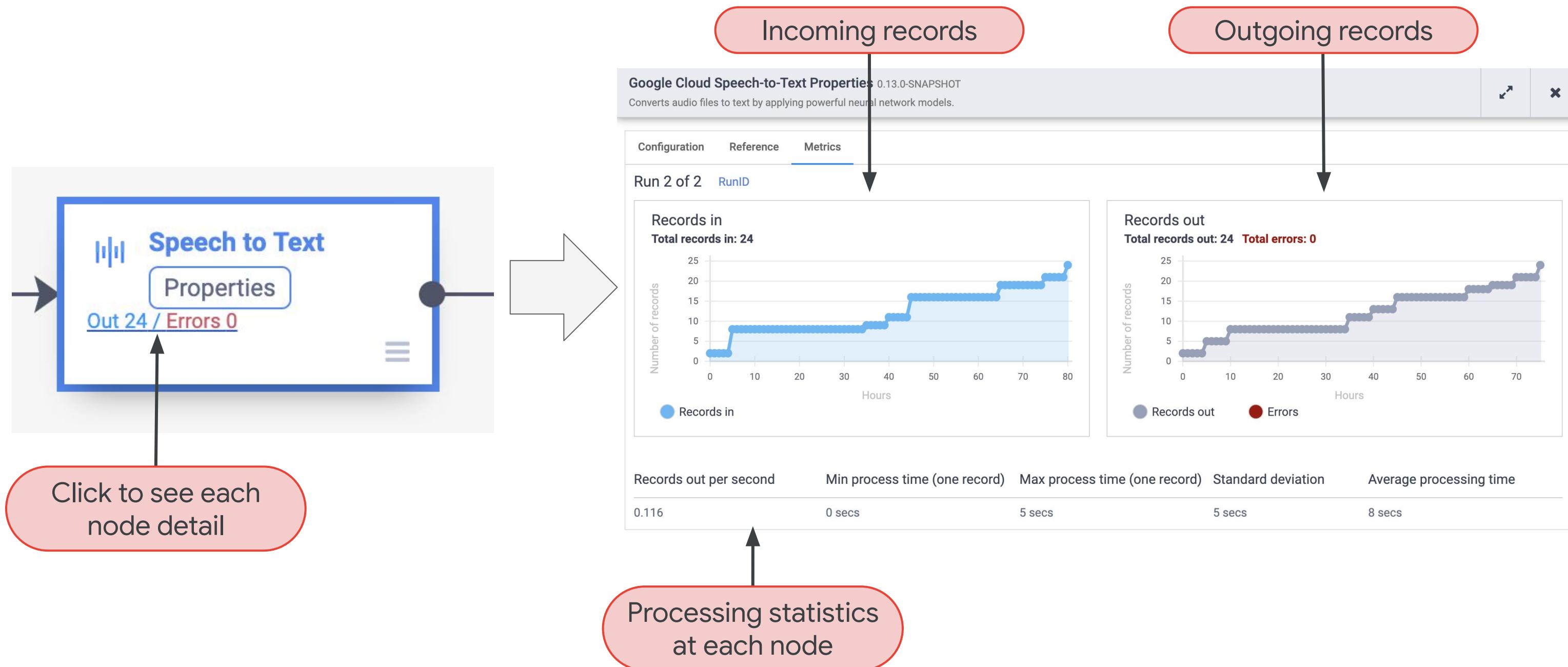
Use Preview Mode to see how the pipeline will run



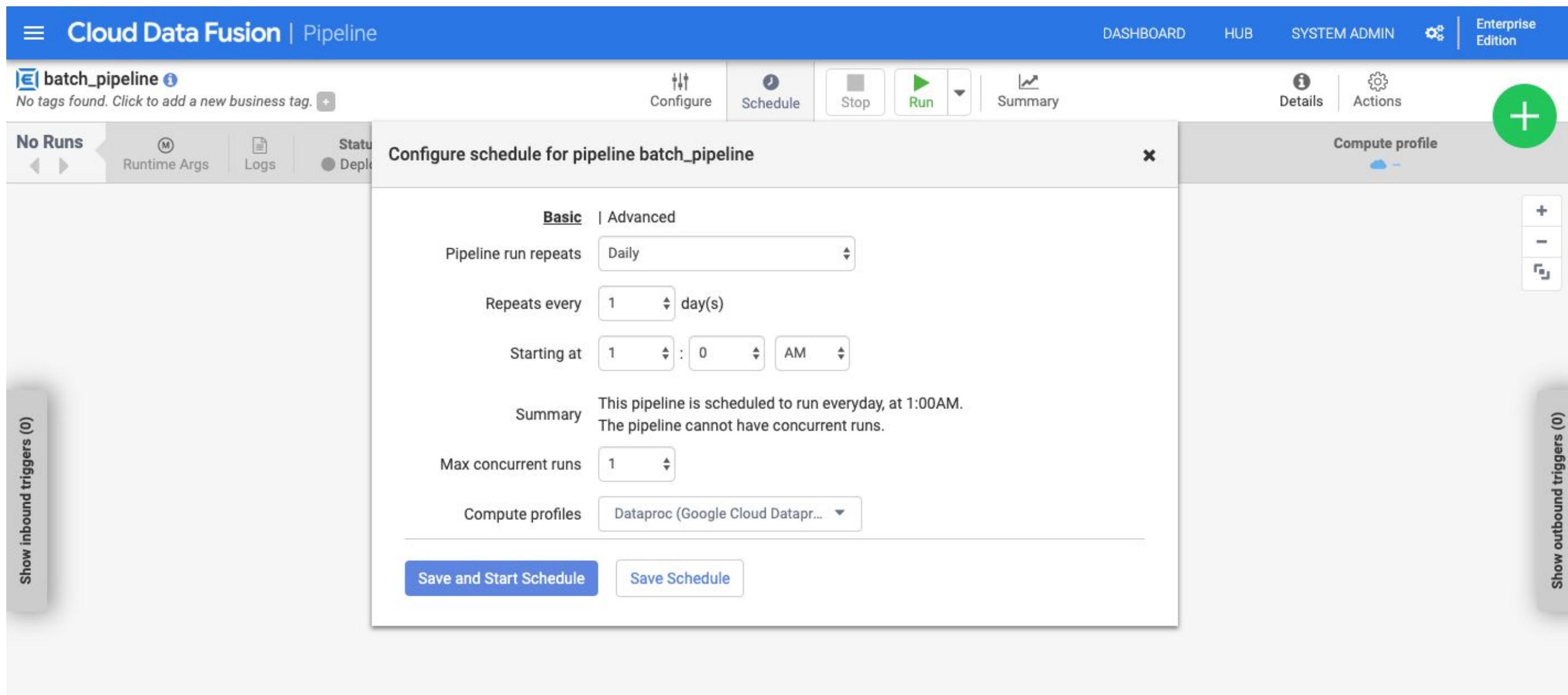
Monitor the health of the entire pipeline



Monitor the health of a single node



You can schedule batch pipelines



After data is transformed, you can track field-level lineage

Relationship between fields of datasets: Provenance, Impact

The screenshot shows the Cloud Data Fusion Field Level Lineage interface. At the top, there's a navigation bar with 'Cloud Data Fusion | Field Level Lineage', 'DASHBOARD', 'HUB', 'SYSTEM ADMIN', and 'Enterprise Edition'. Below the navigation bar, it says 'doubleclick' and 'Dataset'. The main area is titled 'Field level lineage' with the sub-instruction 'Explore root cause and impact for each of the fields of the dataset'. It shows three sections: 'Cause for: doubleclick: campaign', 'Dataset: doubleclick', and 'Impact for: doubleclick: campaign'. The 'Dataset: doubleclick' section lists fields: advertiser_id, timestamp, referrer_url, campaign_id, landing_page_url, advertiser, and landing_page_url_id. Arrows indicate 'Incoming operations' pointing to 'campaign' and 'Outgoing operations' pointing away from 'campaign'. A search bar 'Search by field name' is also present.

Dataset name	Field name
1 campaign	offset
	body

Dataset name	Field name
1 hits	campaign

See every operation that is made on a field

Operations applied to a field

The screenshot shows the Cloud Data Fusion Field Level Lineage interface. The top navigation bar includes 'Cloud Data Fusion | Field Level Lineage', 'DASHBOARD', 'HUB', 'SYSTEM ADMIN', and 'Enterprise Edition'. The main page title is 'doubleclick' under 'Dataset'. A modal window titled 'Cause operations for field 'campaign'' is open, displaying the 'Operations between 'campaign' and 'doubleclick'' section. It shows one operation (1 of 1) last executed on 03-27-2019 at 09:48:28 AM by '00-BigQuery_SQL_DoubleClick_v1'. The table details three operations:

Input	Input fields	Operation	Description	Output fields	Output
1 campaign	--	campaign.Re...	Read from Google Cloud Storage.	offset, body	--
2 --	[offset], [body]	parse campaigns.P... Data	parse-as-csv :body '' true; drop :body;	advertiser_id, campaign_id, campaign	--
3 --	[campaign]	Joiner3.Iden... parse	Unchanged as part of a join campaigns.c...	campaign	--

Data analysts can explore datasets in the Wrangler

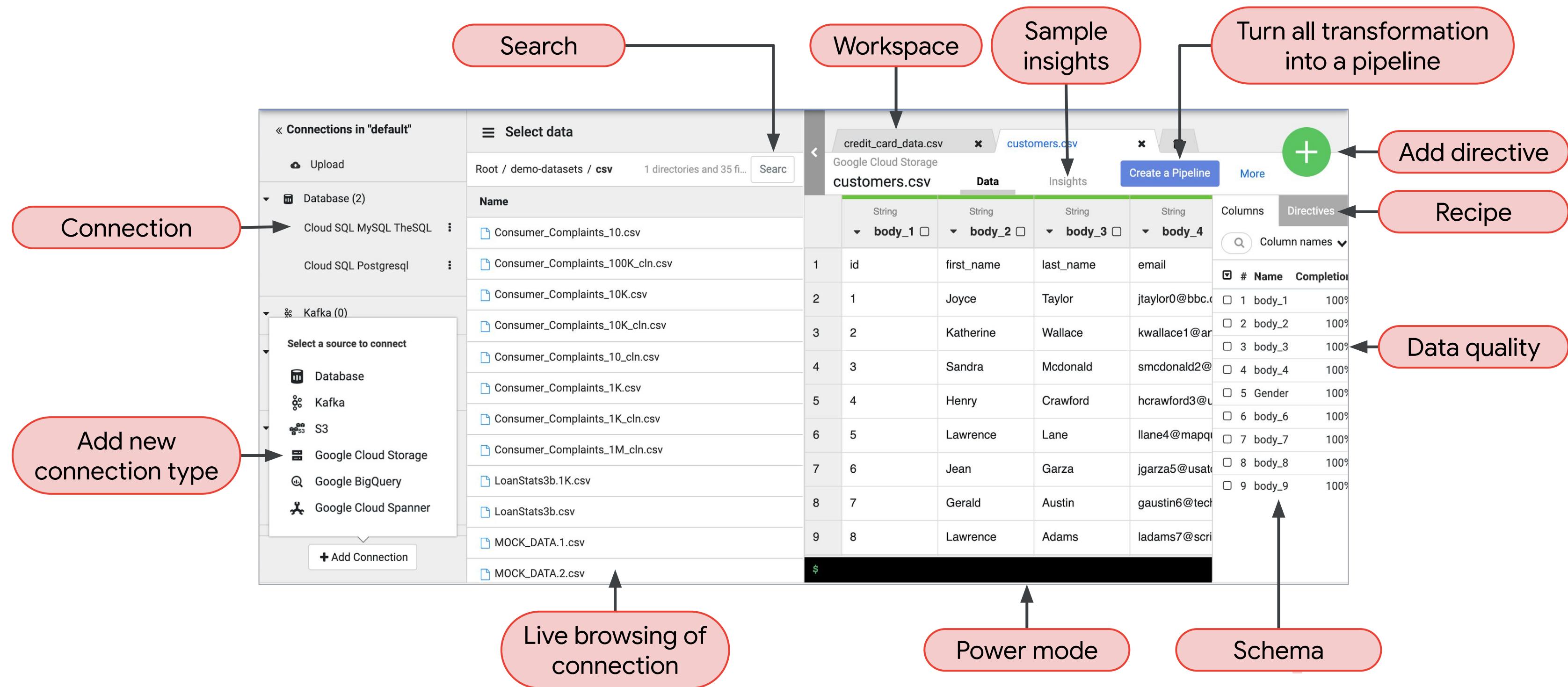
Wrangler is a code-free, visual environment for transforming data in data pipelines.

The screenshot shows the Google Cloud Wrangler interface, which is a visual environment for transforming data in data pipelines. The left sidebar shows a navigation menu with options like Namespace, Control Center, Pipeline, List, Studio, Wrangler (which is selected), and Metadata. The main area is titled "Select data" and shows a pipeline with three stages: "credit_card_data.csv" (Google Cloud Storage), "customers.csv" (Google Cloud Storage), and "customers.csv" (Google Cloud Storage). The middle stage is currently selected. The pipeline interface includes tabs for "Data" and "Insights", a "Create a Pipeline" button, and a "More" button. Below the pipeline, there is a detailed view of the "credit_card_data.csv" dataset, showing columns: id, first_name, last_name, email, credit_card, and credit_card_type. The data table contains 8 rows of sample data. To the right of the table, there is a "Columns (6)" section and a "Directives (4)" section, which lists the following transformation steps:

- # Directives
- 1 parse-as-csv :body "true"
- 2 drop body
- 3 send-to-error !dq:isCreditCard(credit_card)
- 4 filter-rows-on regex-match credit_card_TYP...

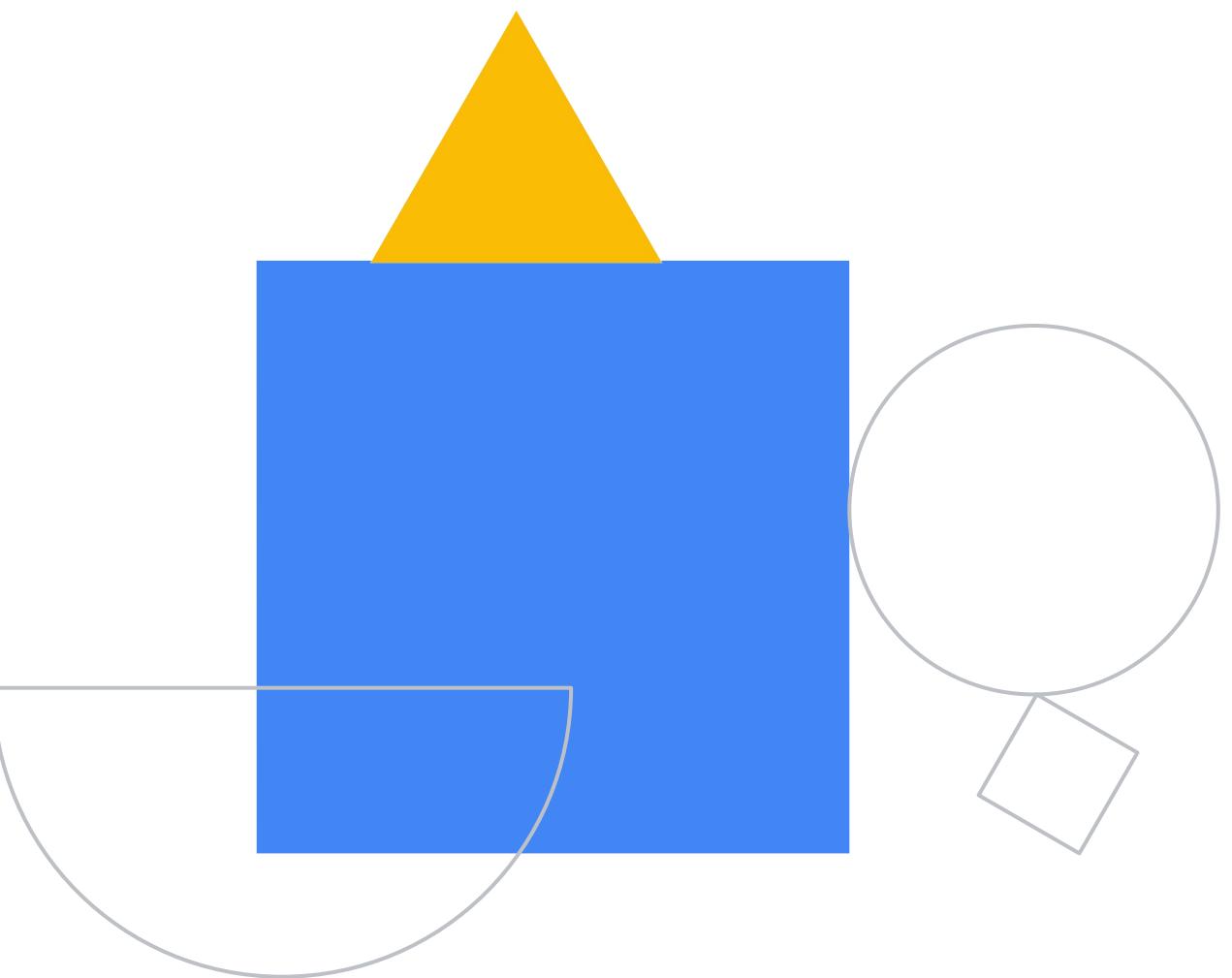
At the bottom right of the interface, it says "Instance Id: cloud-data-fusion-demos/cdf-demo".

Wrangler UI overview for exploring datasets



Lab Intro

Building and Executing a Pipeline
Graph in Cloud Data Fusion



Lab objectives

- 01 Connect Cloud Data Fusion to a couple of data sources
- 02 Apply basic transformations
- 03 Join two data sources
- 04 Write data to a sink



Manage Data Pipelines with Cloud Data Fusion and Cloud Composer

01

- Building batch data pipelines visually with Cloud Data Fusion
- Components
 - UI overview
 - Building a pipeline
 - Exploring data using Wrangler

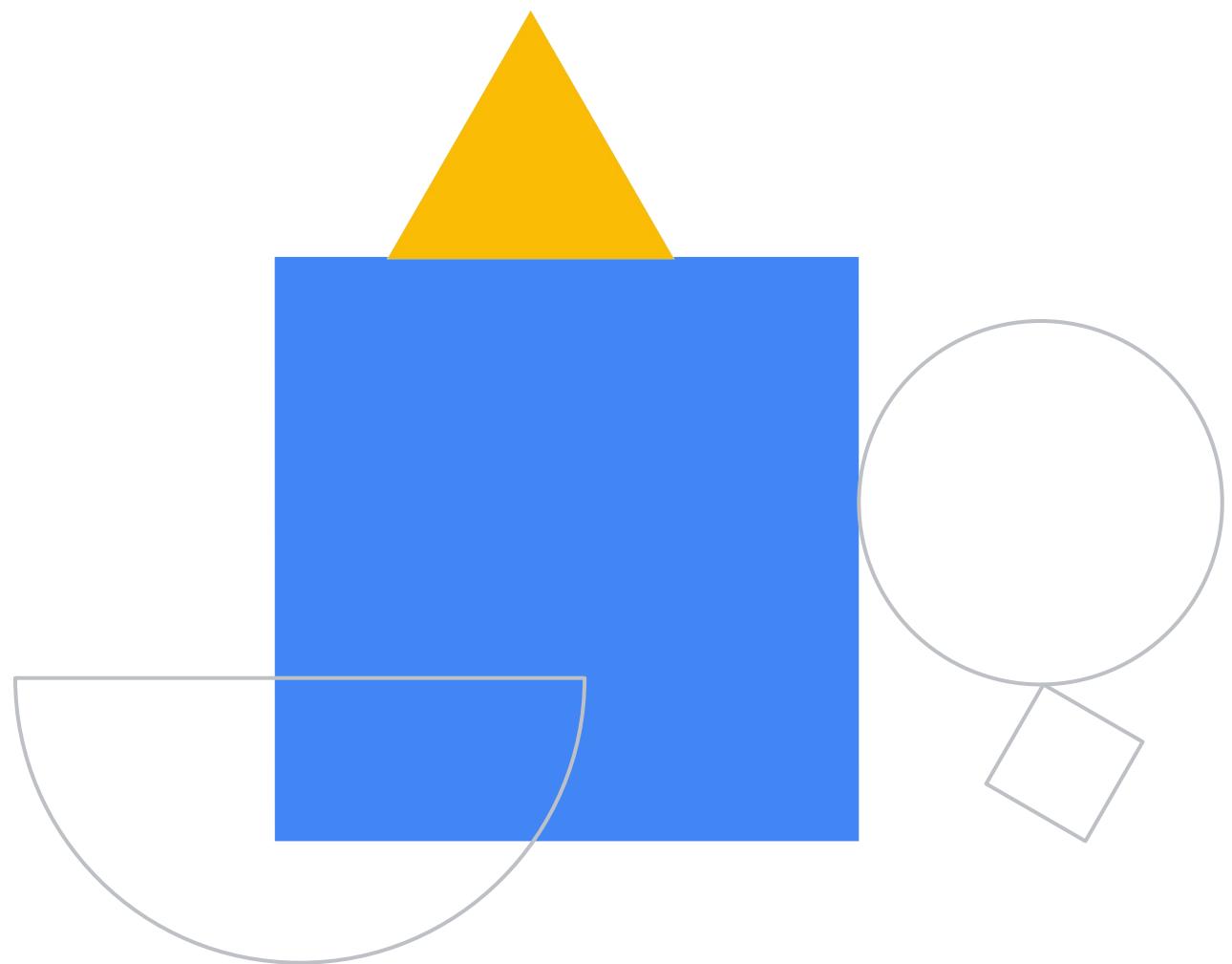
02

- Orchestrating work between Google Cloud services with Cloud Composer
- Apache Airflow environment
 - DAGs and Operators
 - Workflow Scheduling
 - Monitoring and Logging

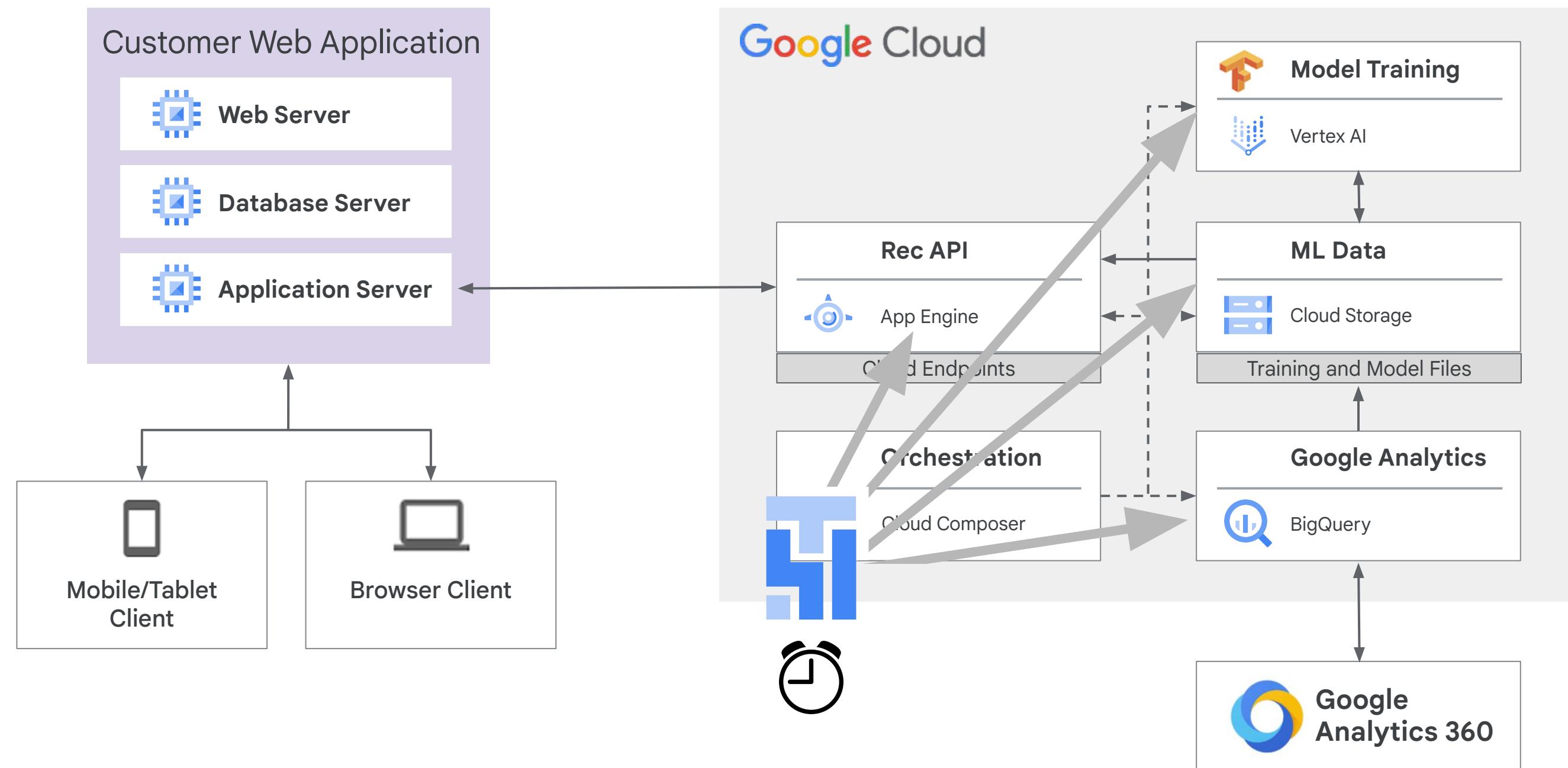


Lab Startup

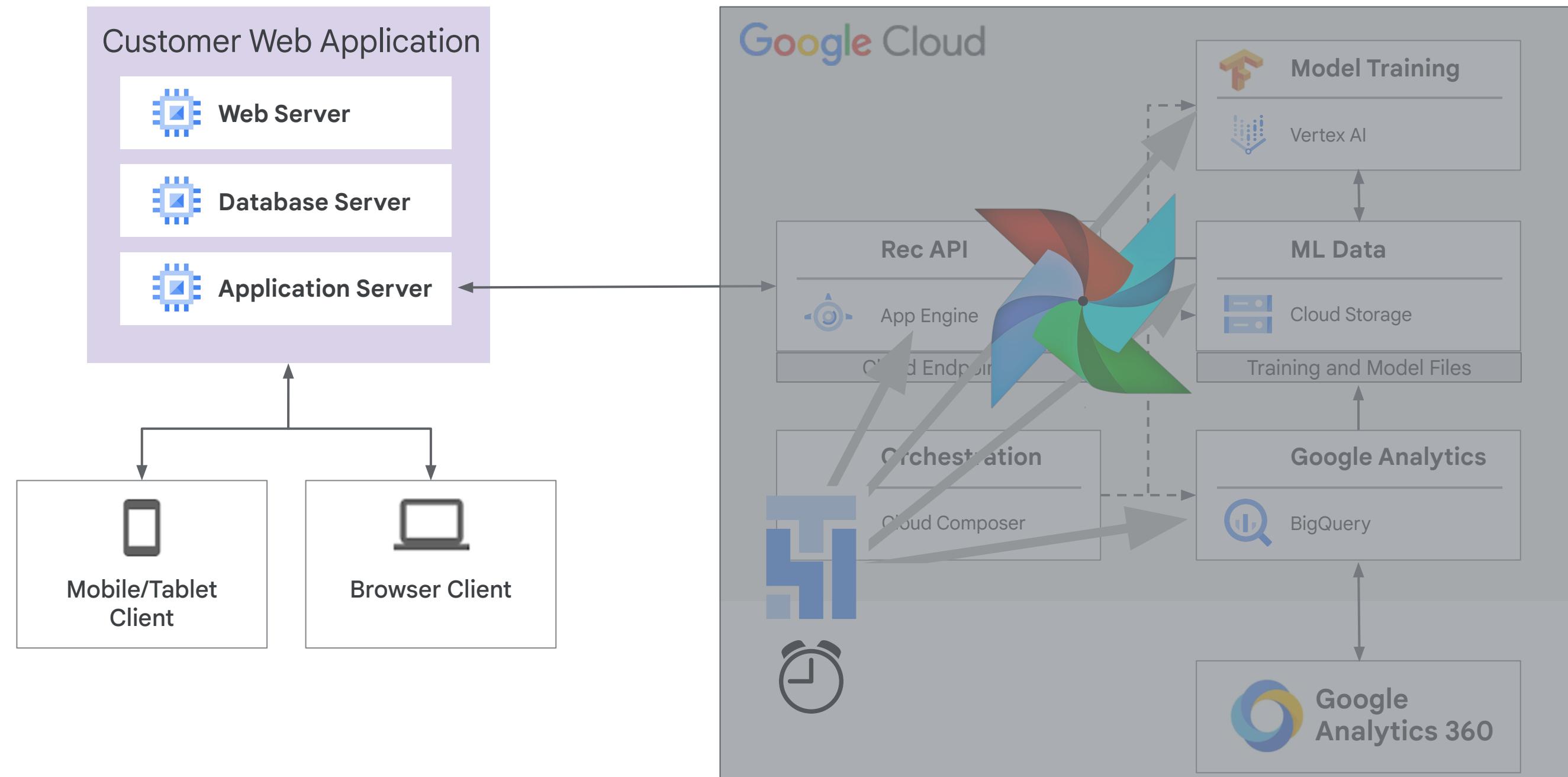
An Introduction to
Cloud Composer



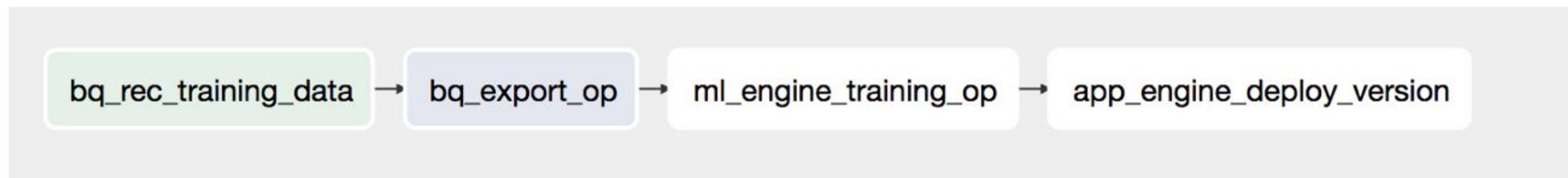
Cloud Composer orchestrates automatic workflows



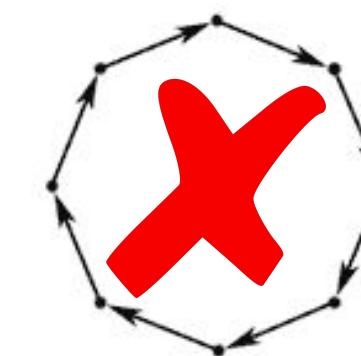
Cloud Composer is managed Apache Airflow



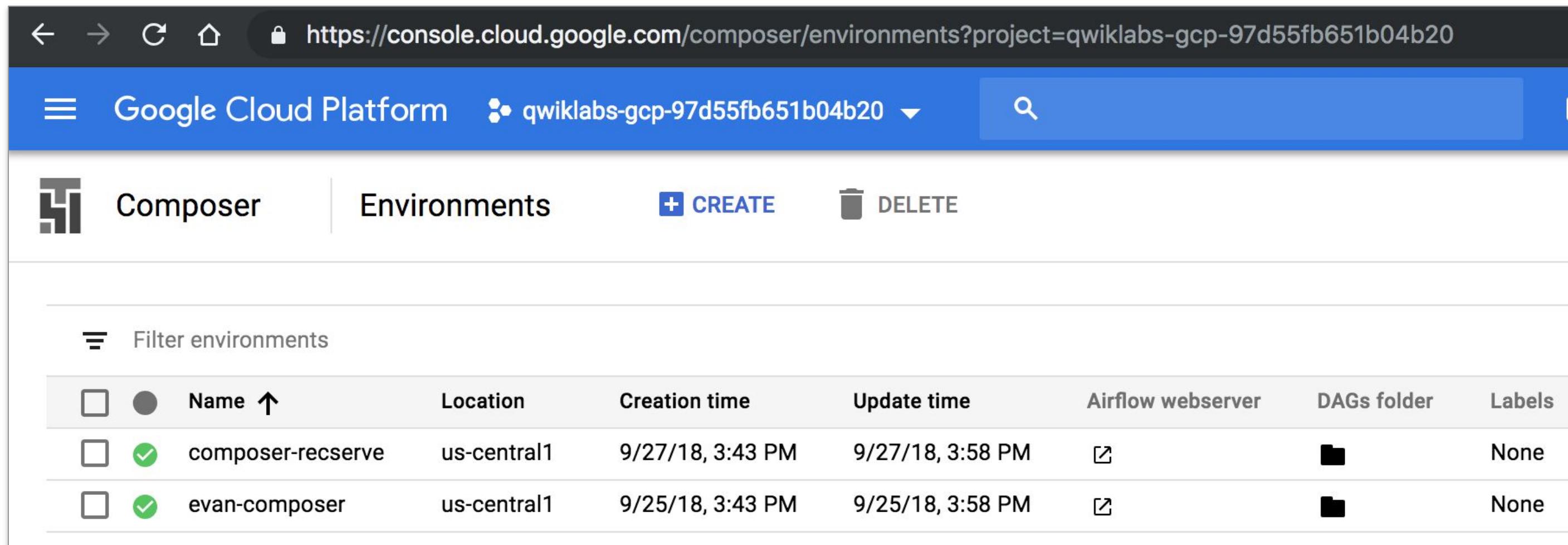
Use Apache Airflow DAGs to orchestrate Google Cloud services



DAG = Directed Acyclic Graph



Cloud Composer creates managed Apache Airflow environments



The screenshot shows the Google Cloud Platform interface for managing Composer environments. The URL in the browser bar is <https://console.cloud.google.com/composer/environments?project=qwiklabs-gcp-97d55fb651b04b20>. The top navigation bar includes the Google Cloud Platform logo, the project name "qwiklabs-gcp-97d55fb651b04b20", and a search bar. Below the navigation is a header with "Composer" (selected), "Environments", "+ CREATE", and "DELETE" buttons. A "Filter environments" section allows sorting by "Name ↑". The main table lists two environments:

	Name	Location	Creation time	Update time	Airflow webserver	DAGs folder	Labels
<input type="checkbox"/>	<input checked="" type="radio"/> composer-recserve	us-central1	9/27/18, 3:43 PM	9/27/18, 3:58 PM			None
<input type="checkbox"/>	<input checked="" type="radio"/> evan-composer	us-central1	9/25/18, 3:43 PM	9/25/18, 3:58 PM			None

Each Airflow environment has a separate web server and folder in Cloud Storage for pipeline DAGs

The screenshot shows the Google Cloud Platform Composer interface. At the top, the URL is https://console.cloud.google.com/composer/environments?project=qwiklabs-gcp-97d55fb651b04b20. The navigation bar includes the Google Cloud Platform logo, the project name qwiklabs-gcp-97d55fb651b04b20, and a search bar. Below the header, there are tabs for 'Composer' and 'Environments', with 'Environments' selected. There are 'CREATE' and 'DELETE' buttons. A table lists environments, with columns for Name, Location, Creation time, Update time, Airflow webserver, DAGs folder, and Labels. Two environments are listed: 'composer-recserve' and 'evan-composer'. The 'Airflow webserver' and 'DAGs folder' columns for both environments are highlighted with a green border and circled with the number 1. The 'Labels' column for both environments shows 'None'. The 'Name' column is sorted by ascending Name.

Name	Location	Creation time	Update time	Airflow webserver	DAGs folder	Labels
composer-recserve	us-central1	9/27/18, 3:43 PM	9/27/18, 3:58 PM			None
evan-composer	us-central1	9/25/18, 3:43 PM	9/25/18, 3:58 PM			None

The DAGs folder is simply a Cloud Storage bucket where you will load your pipeline code

Buckets / us-central1-evan-composer-0e85530c-bucket / dags						
<input type="checkbox"/>	Name	Size	Type	Storage class	Last modified	Public access
<input type="checkbox"/>	dataflow/	—	Folder	—	—	Per object
<input type="checkbox"/>	simple_load_dag.py	6.79 KB	text/x-python-script	Multi-Regional	10/1/18, 1:11 PM	Not public
<input type="checkbox"/>	simple.py	2.51 KB	text/x-python-script	Multi-Regional	10/1/18, 1:10 PM	Not public

Airflow workflows are written in Python

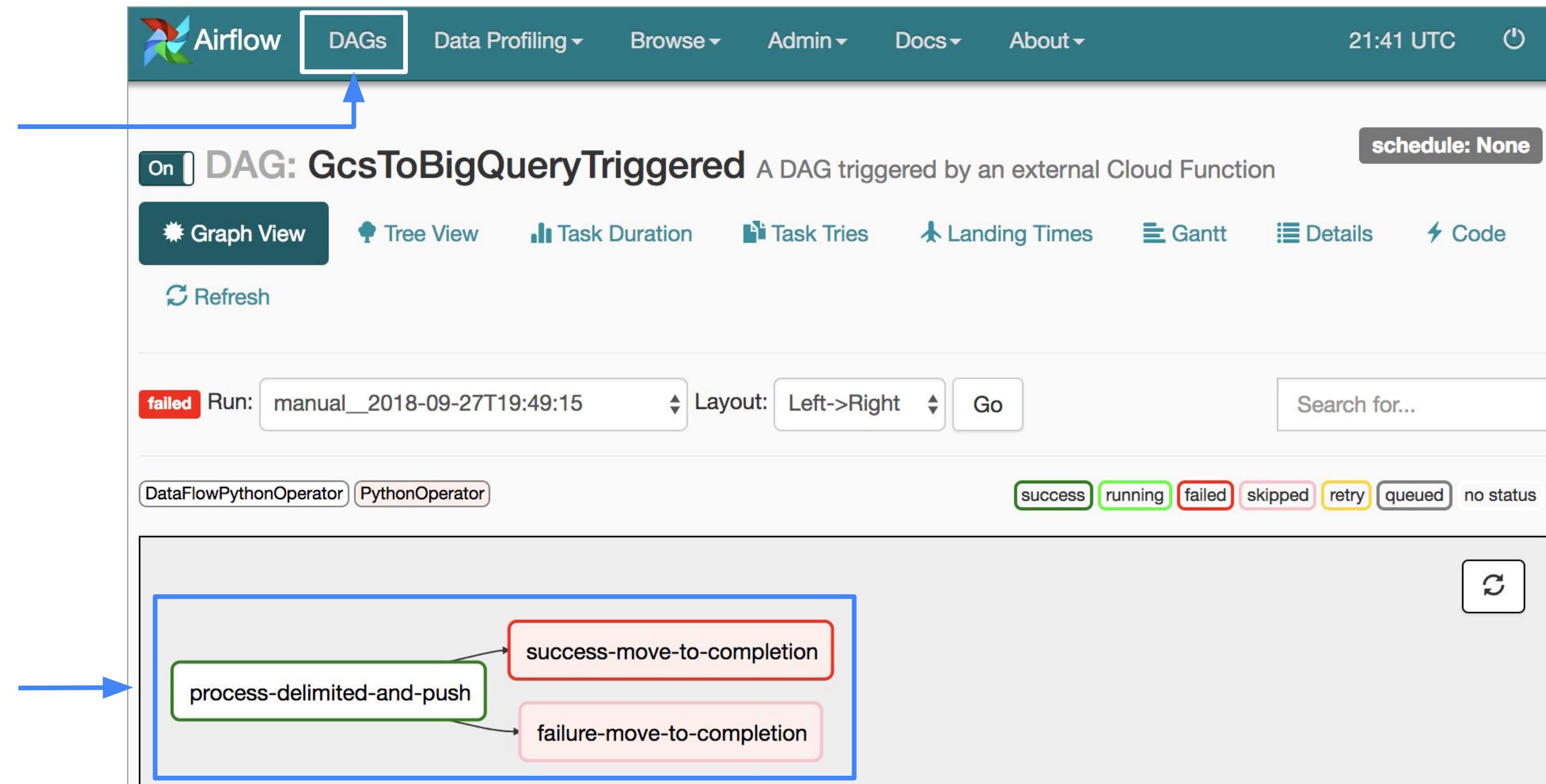
The diagram illustrates the relationship between a Cloud Storage bucket and an Airflow DAG file. On the left, a screenshot of the Google Cloud Storage interface shows a folder named 'dataflow/' containing two files: 'simple_load_dag.py' (6.79 KB, text/x-python-script) and 'simple.py' (2.51 KB, text/x-python-script). A green box highlights 'simple_load_dag.py'. Two green arrows point from this box to the right side of the slide, where the Python code for the DAG is displayed.

```

106 # e.g. state,gender,year,name,number,created_date
107 with models.DAG(dag_id='GcsToBigQueryTriggered',
108     description='A DAG triggered by an external Cloud Function',
109     schedule_interval=None, default_args=DEFAULT_DAG_ARGS) as dag:
110     # Args required for the Dataflow job.
111     job_args = {
112         'input': 'gs://{{ dag_run.conf["bucket"] }}/{{ dag_run.conf["name"] }}',
113         'output': models.Variable.get('bq_output_table'),
114         'fields': models.Variable.get('input_field_names'),
115         'load_dt': DS_TAG
116     }
117
118     # Main Dataflow task that will process and load the input delimited file.
119     dataflow_task = dataflow_operator.DataFlowPythonOperator(
120         task_id="process-delimited-and-push",
121         py_file=DATAFLOW_FILE,
122         options=job_args_
123
124     # Here we create two conditional tasks, one of which will be executed
125     # based on whether the dataflow_task was a success or a failure.
126     success_move_task = python_operator.PythonOperator(task_id='success-move-to-completion',
127             python_callable=move_to_completion_bucket,
128             # A success_tag is used to move
129             # the input file to a success
130             # prefixed folder.
131             op_args=[COMPLETION_BUCKET, SUCCESS_TAG],
132             provide_context=True,
133             trigger_rule=TriggerRule.ALL_SUCCESS)
134
135     failure_move_task = python_operator.PythonOperator(task_id='failure-move-to-completion',
136             python_callable=move_to_completion_bucket,
137             # A failure_tag is used to move
138             # the input file to a failure
139             # prefixed folder.
140             op_args=[COMPLETION_BUCKET, FAILURE_TAG],
141             provide_context=True,
142             trigger_rule=TriggerRule.ALL_FAILED)
143
144     # The success_move_task and failure_move_task are both downstream from the
145     # dataflow_task.
146     dataflow_task >> success_move_task
147     dataflow_task >> failure_move_task

```

The Python file creates a DAG



Airflow web server UI overview

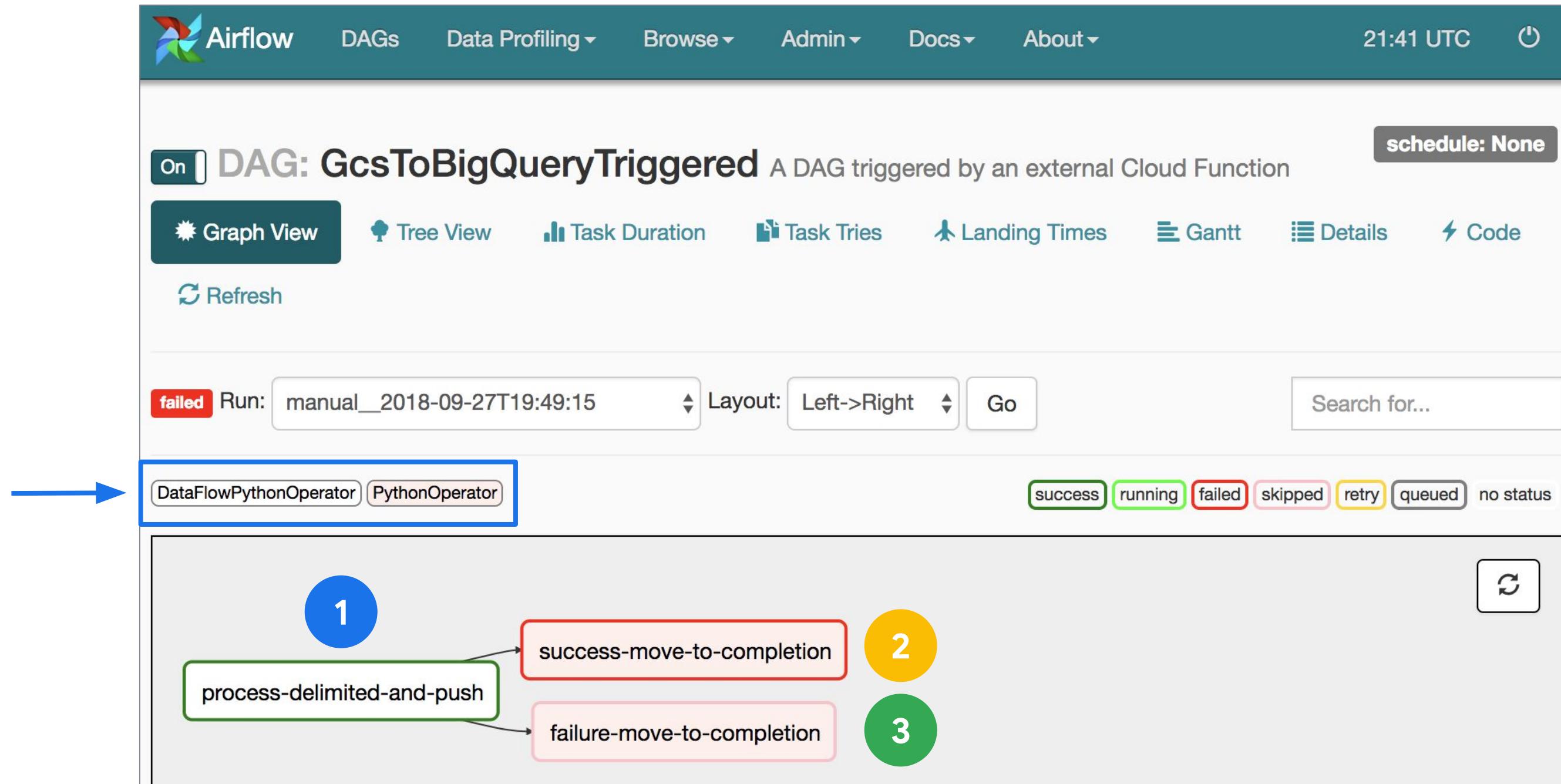
The screenshot shows the Airflow web server UI for the DAG: GcsToBigQueryTriggered. The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, About, the current time (21:41 UTC), and a power icon.

The main header displays the DAG name "DAG: GcsToBigQueryTriggered" with a subtitle "A DAG triggered by an external Cloud Function" and a status "schedule: None". Below the header are several navigation buttons: Graph View (selected), Tree View, Task Duration, Task Tries, Landing Times, Gantt, Details, and Code. A "Refresh" button is also present.

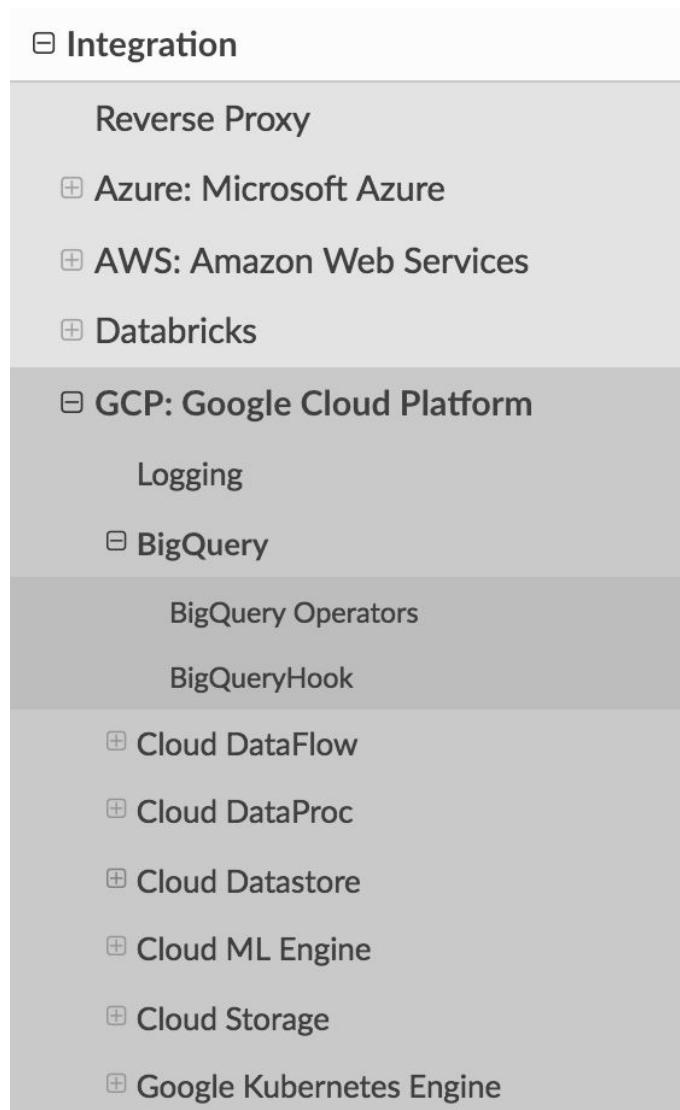
Below the navigation is a search bar with the text "Search for...". Underneath the search bar are two rows of buttons: "Run:" (failed) manual_2018-09-27T19:49:15, "Layout:" (Left->Right), "Go", and a "Search for..." input field. The second row contains buttons for "DataFlowPythonOperator" and "PythonOperator", and a status legend: success (green), running (yellow-green), failed (red), skipped (pink), retry (orange), queued (grey), and no status (grey).

The central area displays the DAG's task graph. A task named "process-delimited-and-push" is highlighted with a green border. It has two outgoing edges: one to a red box labeled "success-move-to-completion" and another to a pink box labeled "failure-move-to-completion". A refresh icon is located in the top right corner of the graph area.

The Python file creates a DAG



Airflow uses operators in your DAG to orchestrate other Google Cloud services

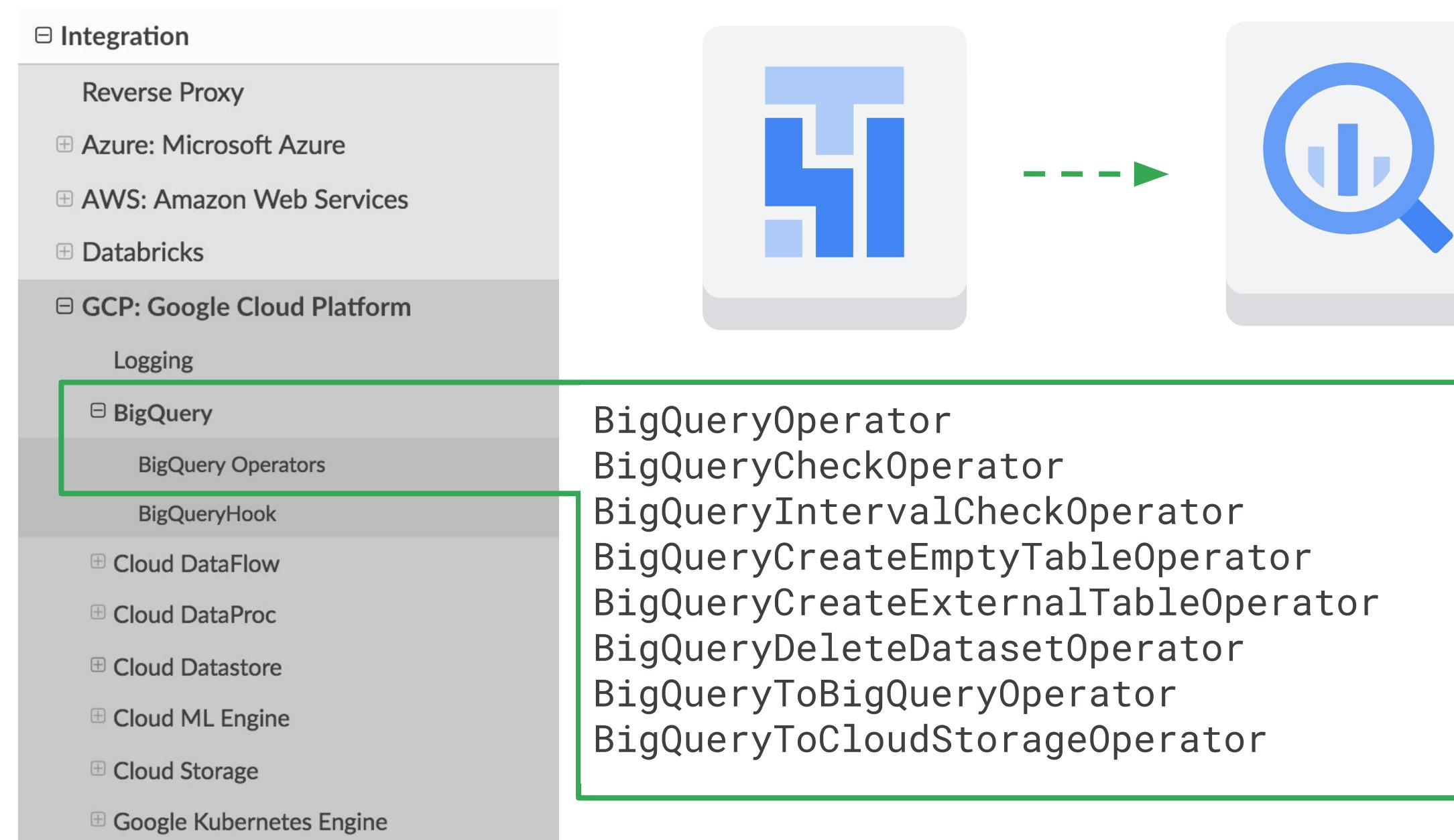


View all Google Cloud services that Airflow can orchestrate:

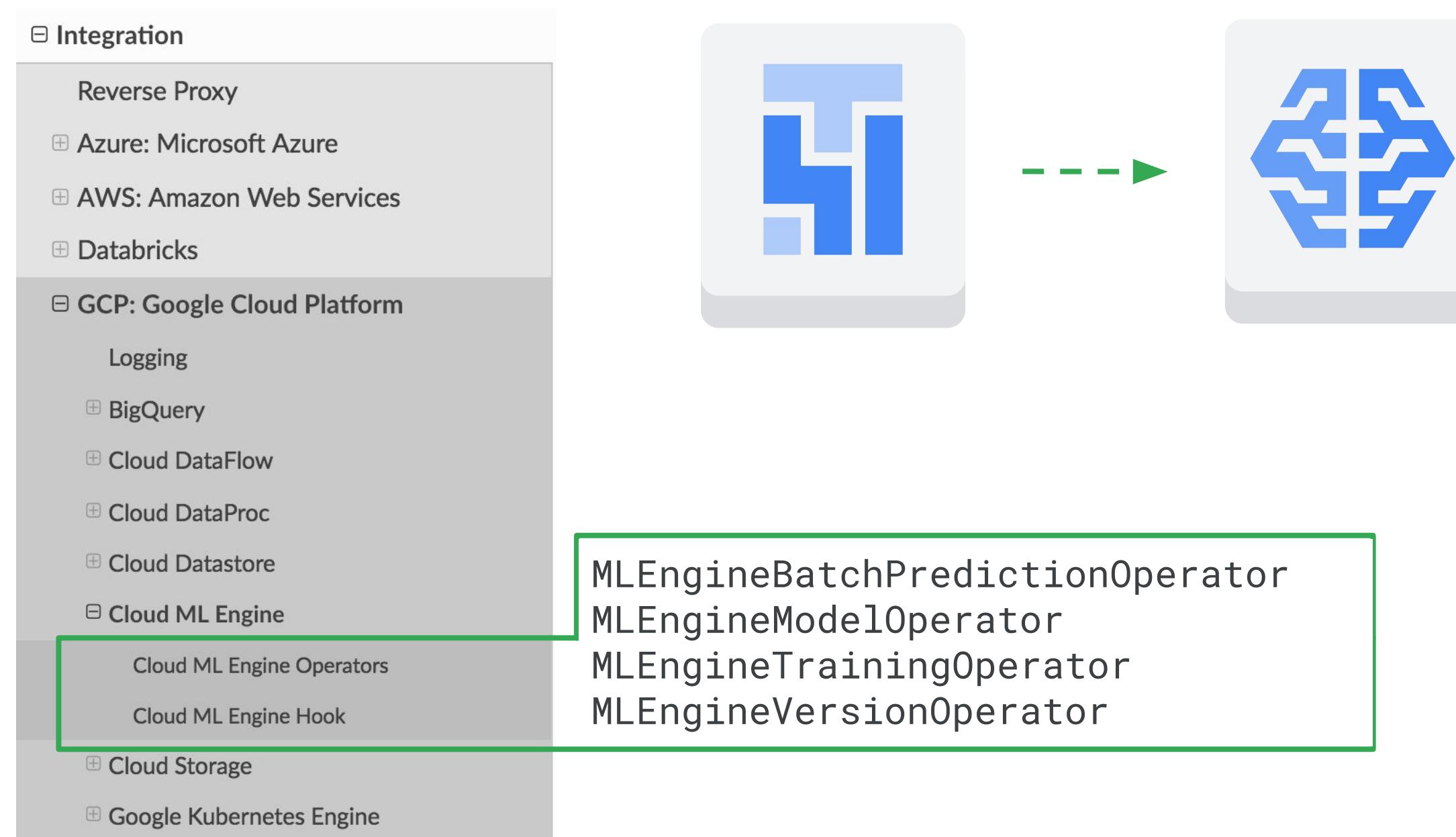
<http://airflow.apache.org/integration.html#gcp>

+ sample code

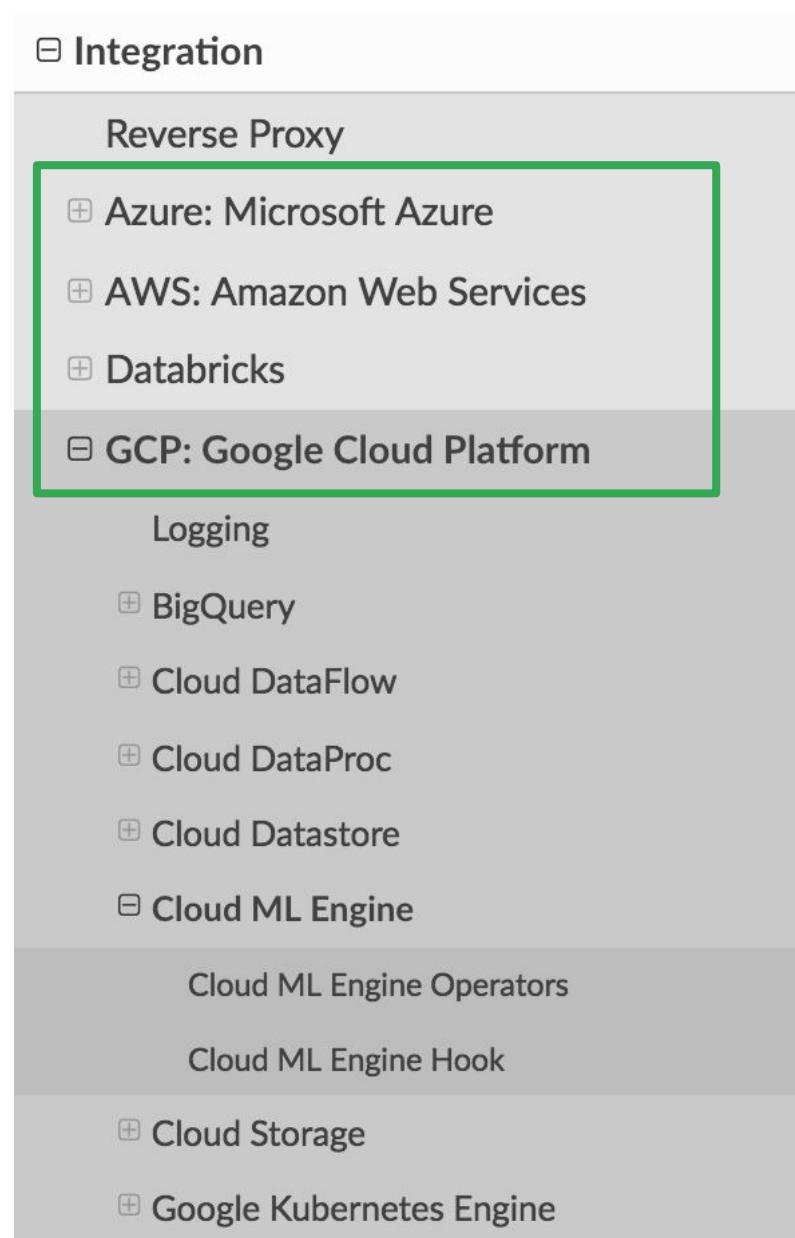
BigQuery Operators are popular for updating your ML training dataset



Vertex AI (formerly Cloud ML Engine) operators can launch training and deployment jobs

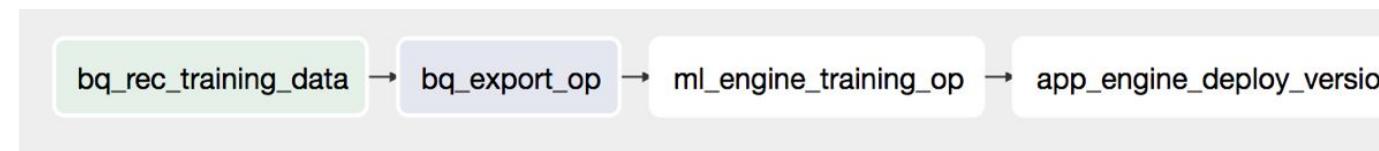


Apache Airflow is open source and cross-platform for hybrid pipelines



Example: Common Machine Learning workflow

DAG Operators



```
# update training data  
t1 = BigQueryOperator(  
)  
  
# BigQuery training data export to GCS  
t2 = BigQueryToCloudStorageOperator(  
)  
  
# AI Platform training job  
t3 = MLEngineTrainingOperator(  
)  
  
# App Engine deploy new version  
t4 = AppEngineVersionOperator(  
)  
  
# DAG dependencies  
t2.set_upstream(t1)  
t3.set_upstream(t2)  
t4.set_upstream(t3)
```

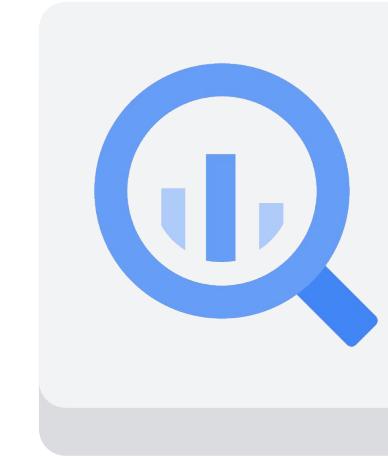
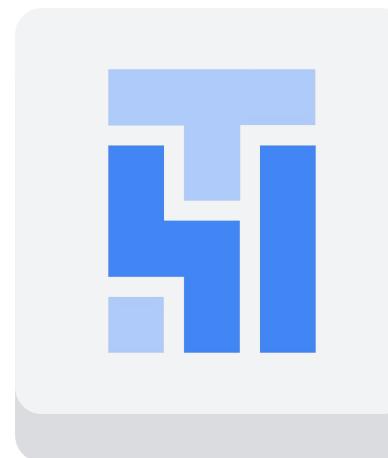
1

2

3

4

BigQuery and Cloud Storage operators get us fresh training data



```
# update training data  
t1 = BigQueryOperator(  
)  
  
# BigQuery training data export to GCS  
t2 = BigQueryToCloudStorageOperator(  
)  
  
# AI Platform training job  
t3 = MLEngineTrainingOperator(  
)  
  
# App Engine deploy new version  
t4 = AppEngineVersionOperator(  
)  
  
# DAG dependencies  
t2.set_upstream(t1)  
t3.set_upstream(t2)  
t4.set_upstream(t3)
```

Use the BigQueryOperator to run SQL

```
from airflow.contrib.operators import bigquery_operator

# constants or can be dynamic based on Airflow macros
max_query_date = '2018-02-01'
min_query_date = '2018-01-01'

# Query recent StackOverflow questions.
bq_recent_questions_query = bigquery_operator.BigQueryOperator(
    task_id='bq_recent_questions_query',
    bq1="""
        SELECT owner_display_name, title, view_count
        FROM `bigquery-public-data.stackoverflow.posts_questions`
        WHERE creation_date < CAST('{max_date}' AS TIMESTAMP)
            AND creation_date >= CAST('{min_date}' AS TIMESTAMP)
        ORDER BY view_count DESC
        LIMIT 100
    """ .format(max_date=max_query_date, min_date=min_query_date),
    use_legacy_sql=False,
    destination_dataset_table=bq_recent_questions_table_id)
```

SQL commands can hold parameters (from Python)

```
from airflow.contrib.operators import bigquery_operator

# constants or can be dynamic based on Airflow macros
max_query_date = '2018-02-01'
min_query_date = '2018-01-01'

# Query recent StackOverflow questions.
bq_recent_questions_query = bigquery_operator.BigQueryOperator(
    task_id='bq_recent_questions_query',
    bq1="""
        SELECT owner_display_name, title, view_count
        FROM `bigquery-public-data.stackoverflow.posts_questions`
        WHERE creation_date < CAST('{max_date}' AS TIMESTAMP)
            AND creation_date >= CAST('{min_date}' AS TIMESTAMP)
        ORDER BY view_count DESC
        LIMIT 100
    """.format(max_date=max_query_date, min_date=min_query_date),
    use_legacy_sql=False,
    destination_dataset_table=bq_recent_questions_table_id)
```

Note the Python constants for a date range here

```
from airflow.contrib.operators import bigquery_operator

# constants or can be dynamic based on Airflow macros
max_query_date = '2018-02-01'
min_query_date = '2018-01-01'

# Query recent StackOverflow questions.
bq_recent_questions_query = bigquery_operator.BigQueryOperator(
    task_id='bq_recent_questions_query',
    bqj="""
        SELECT owner_display_name, title, view_count
        FROM `bigquery-public-data.stackoverflow.posts_questions`
        WHERE creation_date < CAST('{max_date}' AS TIMESTAMP)
            AND creation_date >= CAST('{min_date}' AS TIMESTAMP)
        ORDER BY view_count DESC
        LIMIT 100
    """.format(max_date=max_query_date, min_date=min_query_date),
    use_legacy_sql=False,
    destination_dataset_table=bq_recent_questions_table_id)
```

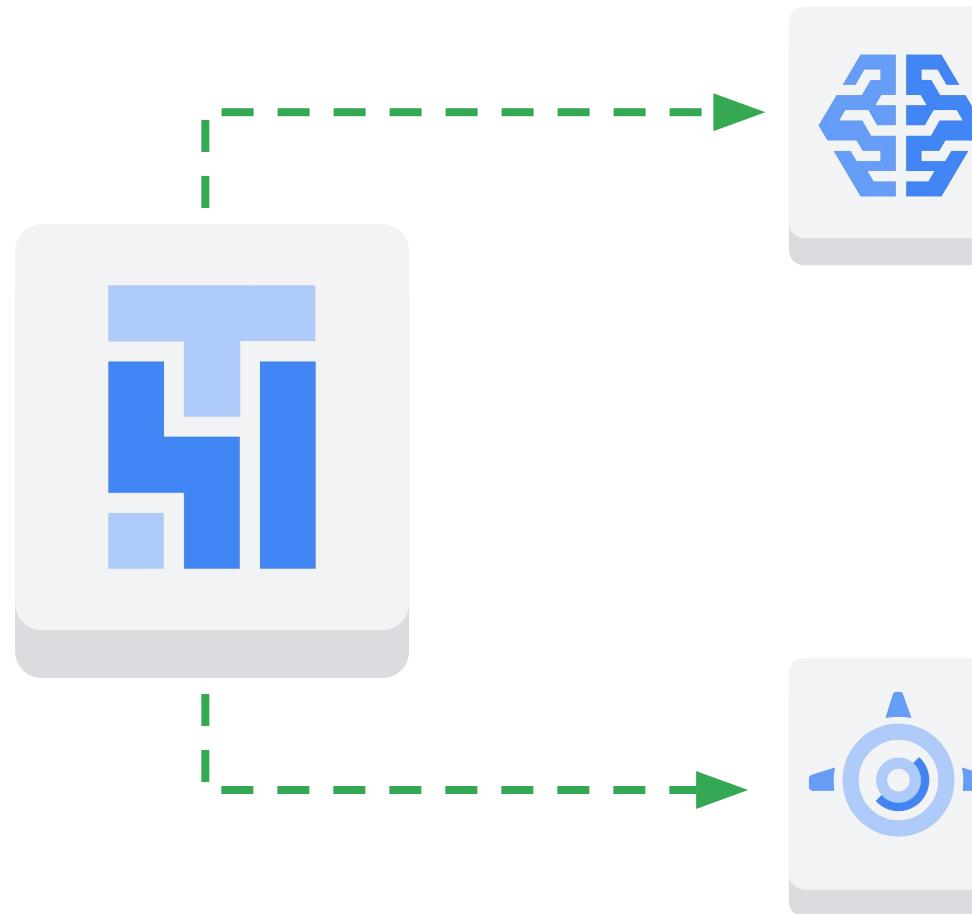
You could even set a rolling window with macros

```
from airflow.contrib.operators import bigquery_operator

# constants or can be dynamic based on Airflow macros
max_query_date = {{ macros.ds_add(ds, -1) }} # one day prior
min_query_date = {{ macros.ds_add(ds, -7) }} # seven days prior

# Query recent StackOverflow questions.
bq_recent_questions_query = bigquery_operator.BigQueryOperator(
    task_id='bq_recent_questions_query',
    bqj="""
        SELECT owner_display_name, title, view_count
        FROM `bigquery-public-data.stackoverflow.posts_questions`
        WHERE creation_date < CAST('{{max_date}}' AS TIMESTAMP)
            AND creation_date >= CAST('{{min_date}}' AS TIMESTAMP)
        ORDER BY view_count DESC
        LIMIT 100
    """ .format(max_date=max_query_date, min_date=min_query_date),
    use_legacy_sql=False,
    destination_dataset_table=bq_recent_questions_table_id)
```

Vertex AI and App Engine operators retrain and redeploy our model



```
# update training data  
t1 = BigQueryOperator()  
  
# BigQuery training data export to GCS  
t2 = BigQueryToCloudStorageOperator()  
  
# AI Platform training job  
t3 = MLEngineTrainingOperator()  
  
# App Engine deploy new version  
t4 = AppEngineVersionOperator()  
  
# DAG dependencies  
t2.set_upstream(t1)  
t3.set_upstream(t2)  
t4.set_upstream(t3)
```

Manage pipelines and dependencies as code

```
# update training data
t1 = BigQueryOperator(
    )

# BigQuery training data export to GCS
t2 = BigQueryToCloudStorageOperator(
    )

# AI Platform training job
t3 = MLEngineTrainingOperator(
    )

# App Engine deploy new version
t4 = AppEngineVersionOperator(
    )

# DAG dependencies
t2.set_upstream(t1)
t3.set_upstream(t2)
t4.set_upstream(t3)
```

Two scheduling options for Cloud Composer workflows



Periodic



Event-driven

Airflow scheduling basics

Screenshot of the Airflow web interface showing the DAGs page.

The top navigation bar includes: Airflow logo, DAGs, Data Profiling, Browse, Admin, Docs, About, 20:14 UTC, and a power icon.

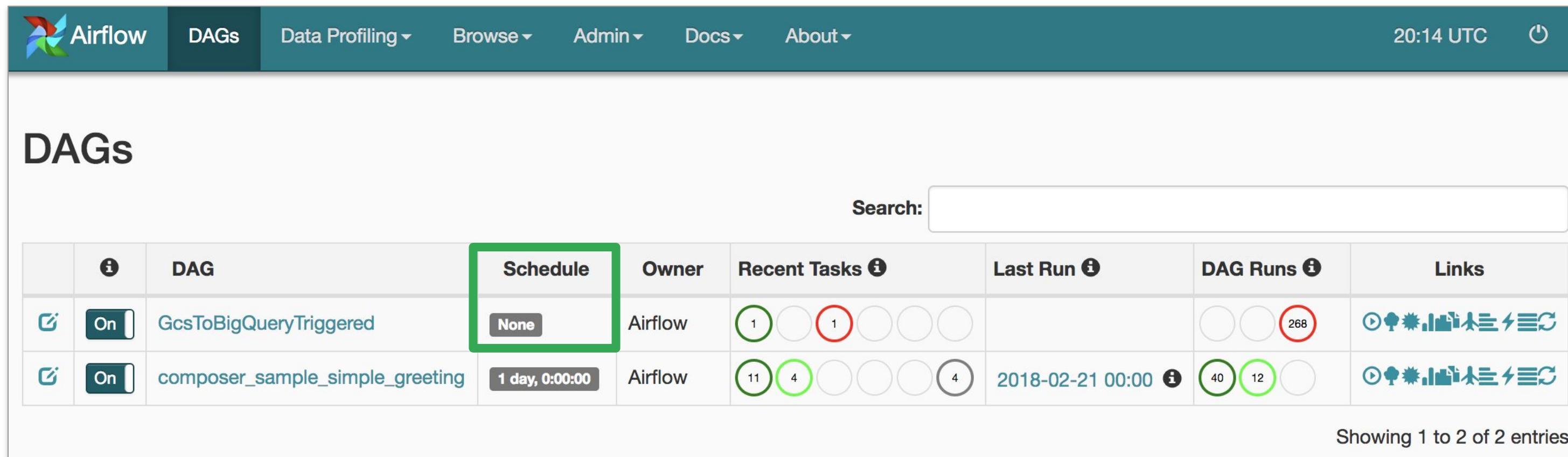
DAGs

Search:

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
		GcsToBigQueryTriggered	None	Airflow				
		composer_sample_simple_greeting	1 day, 0:00:00	Airflow		2018-02-21 00:00		

Showing 1 to 2 of 2 entries

Why is this DAG missing a schedule?



The screenshot shows the Airflow web interface with the following details:

Header: Airflow, DAGs, Data Profiling ▾, Browse ▾, Admin ▾, Docs ▾, About ▾, 20:14 UTC, Power icon.

Section: DAGs

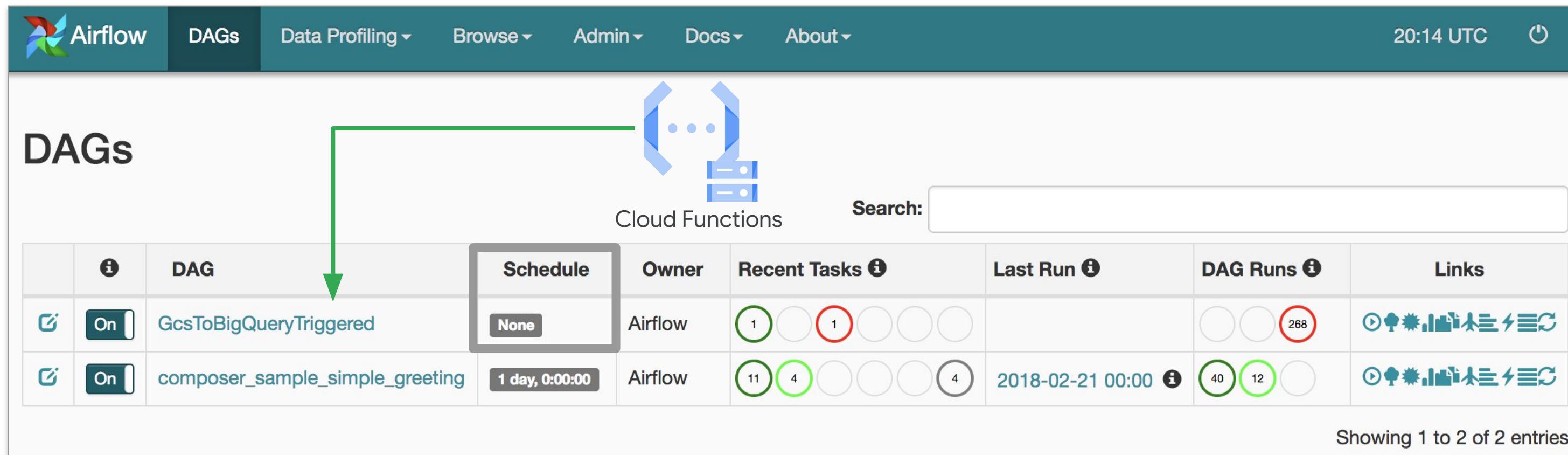
Search: Search bar.

Table: A list of DAGs with the following columns:

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
		GcsToBigQueryTriggered	None	Airflow				
		composer_sample_simple_greeting	1 day, 0:00:00	Airflow		2018-02-21 00:00		

Text: Showing 1 to 2 of 2 entries

Option 1: Event-driven scheduling with Cloud Functions



The screenshot shows the Airflow web interface with the following details:

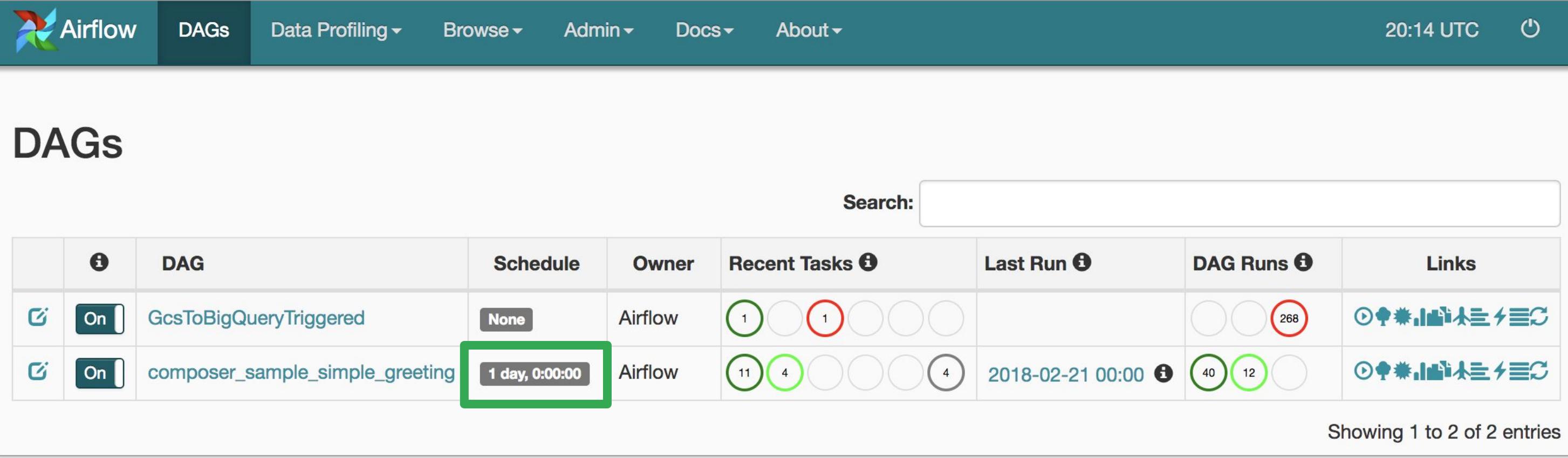
DAGs (Section):

- Cloud Functions** icon: A blue icon representing Cloud Functions.
- Search:** A search bar.

	i	DAG	Schedule	Owner	Recent Tasks <small>i</small>	Last Run <small>i</small>	DAG Runs <small>i</small>	Links
<input checked="" type="checkbox"/>	On	GcsToBigQueryTriggered	None	Airflow	1 1 1 1 1		268
<input checked="" type="checkbox"/>	On	composer_sample_simple_greeting	1 day, 0:00:00	Airflow	11 4 4 4 4	2018-02-21 00:00 <small>i</small>	40 12 12

Showing 1 to 2 of 2 entries

Option 2: Specify pipeline schedule_interval in your DAG



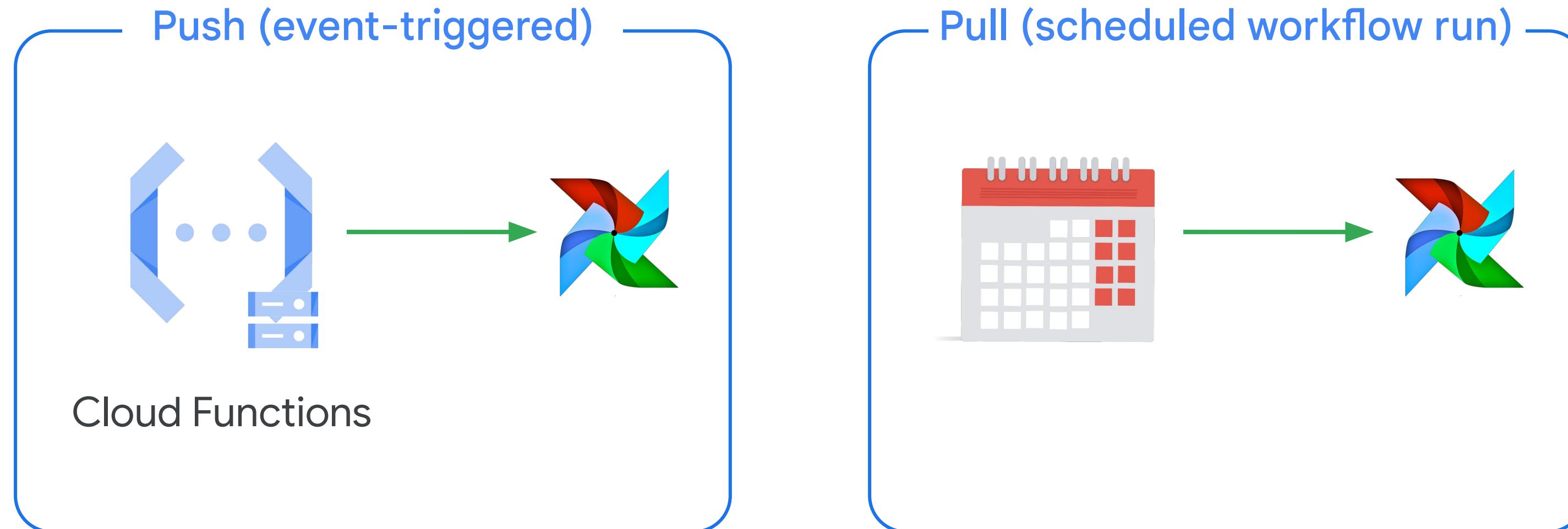
The screenshot shows the Airflow web interface with the 'DAGs' tab selected. The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, About, the current time (20:14 UTC), and a power icon. Below the navigation is a search bar labeled 'Search:'. The main content area displays a table of DAGs. The first row shows 'GcsToBigQueryTriggered' with a 'Schedule' of 'None'. The second row shows 'composer_sample_simple_greeting' with a 'Schedule' of '1 day, 0:00:00', which is highlighted with a green box. Both rows show the owner as 'Airflow'. The 'Recent Tasks' column shows task counts: 1 (green circle), 1 (red circle), and 4 (green circles). The 'Last Run' column shows the date '2018-02-21 00:00' with an info icon. The 'DAG Runs' column shows counts: 268 (red circle) and 40 (green circle), 12 (green circle), and 0 (white circle). The 'Links' column contains icons for Data Profiling, Task Instances, DAG Runs, and Log.

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
		GcsToBigQueryTriggered	None	Airflow				
		composer_sample_simple_greeting	1 day, 0:00:00	Airflow		2018-02-21 00:00		

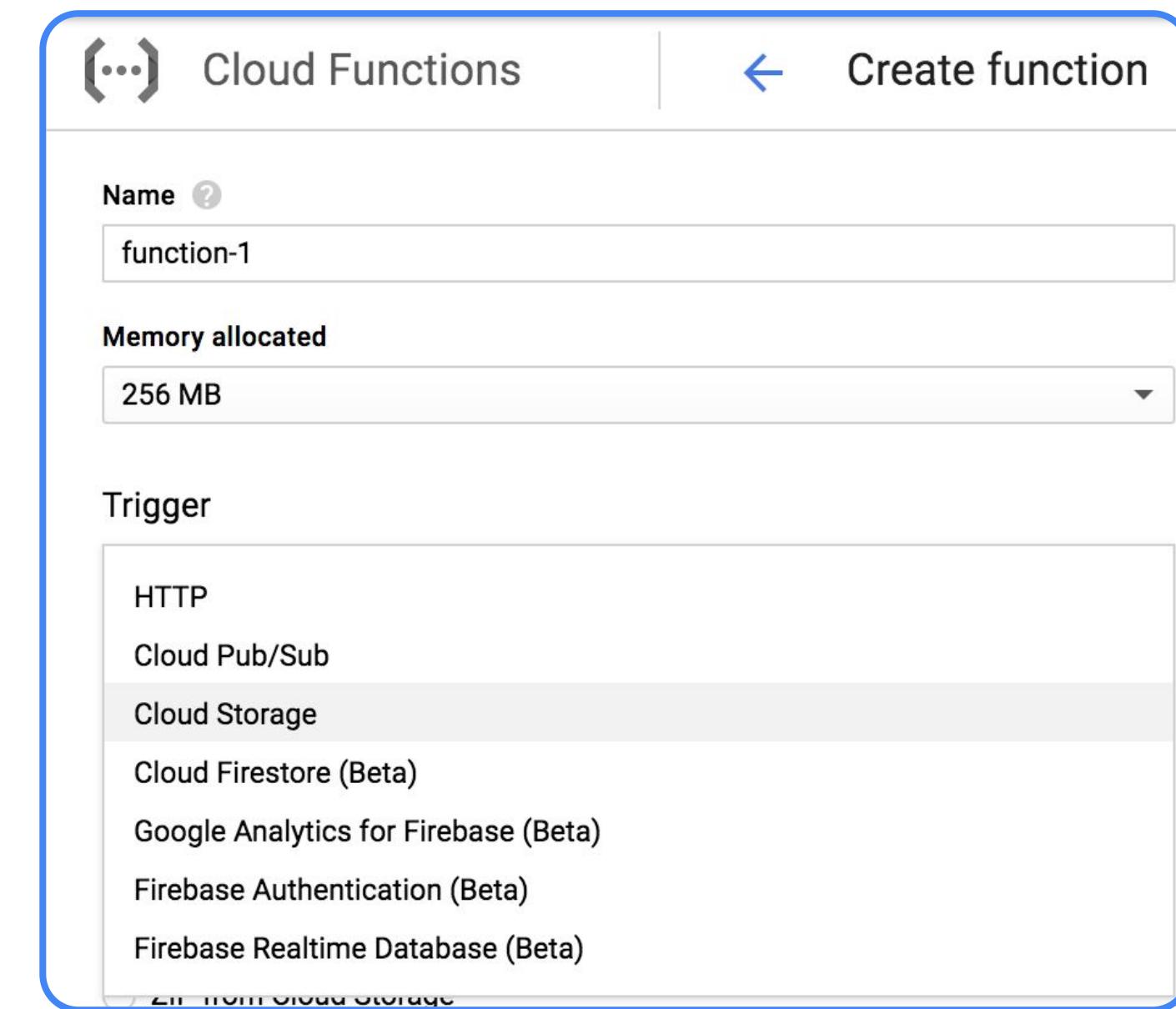
Showing 1 to 2 of 2 entries

```
with models.DAG(  
    'composer_sample_simple_greeting',  
    schedule_interval=datetime.timedelta(days=1),  
    default_args=default_dag_args) as dag:
```

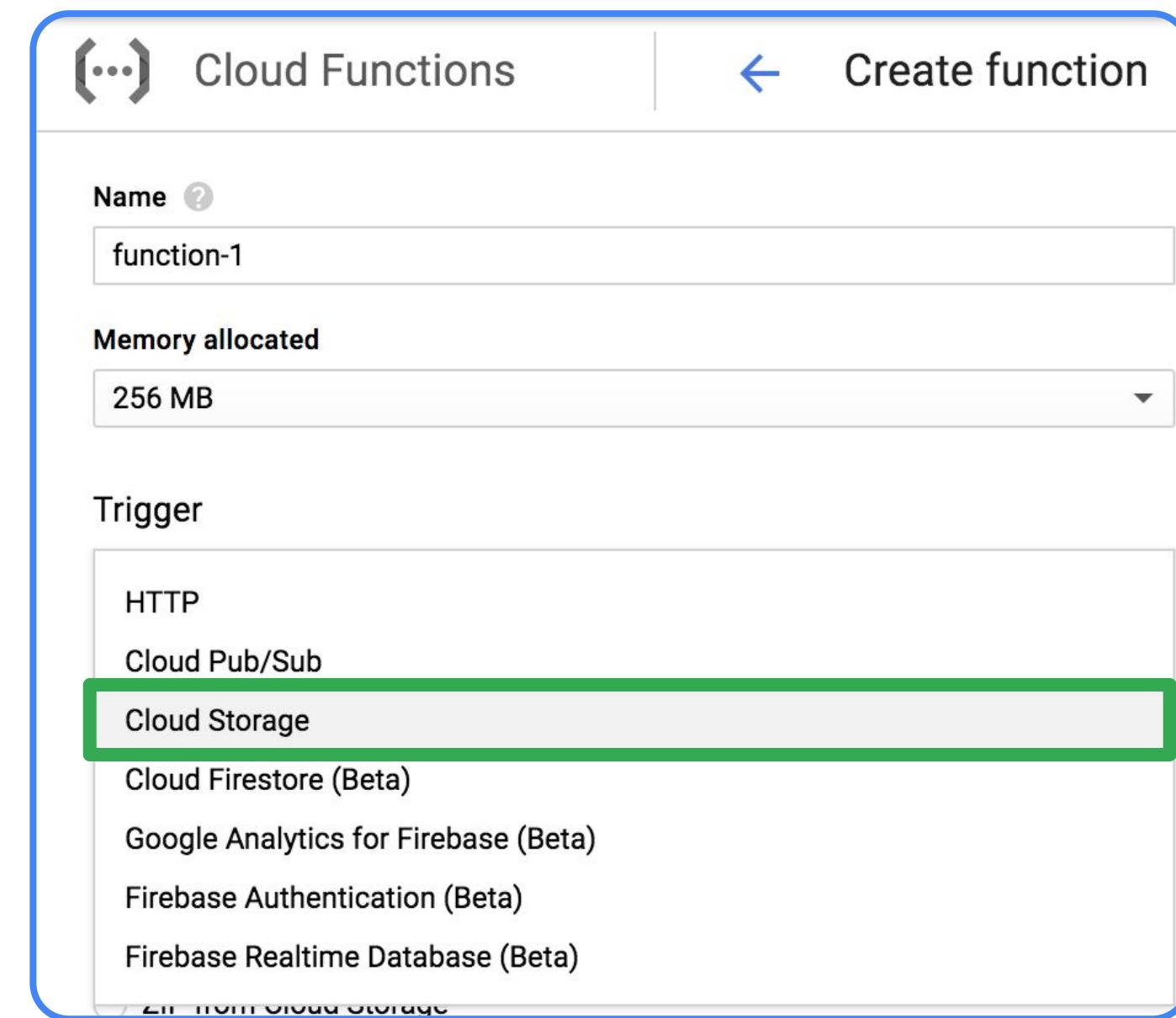
Two types of workflow ETL patterns



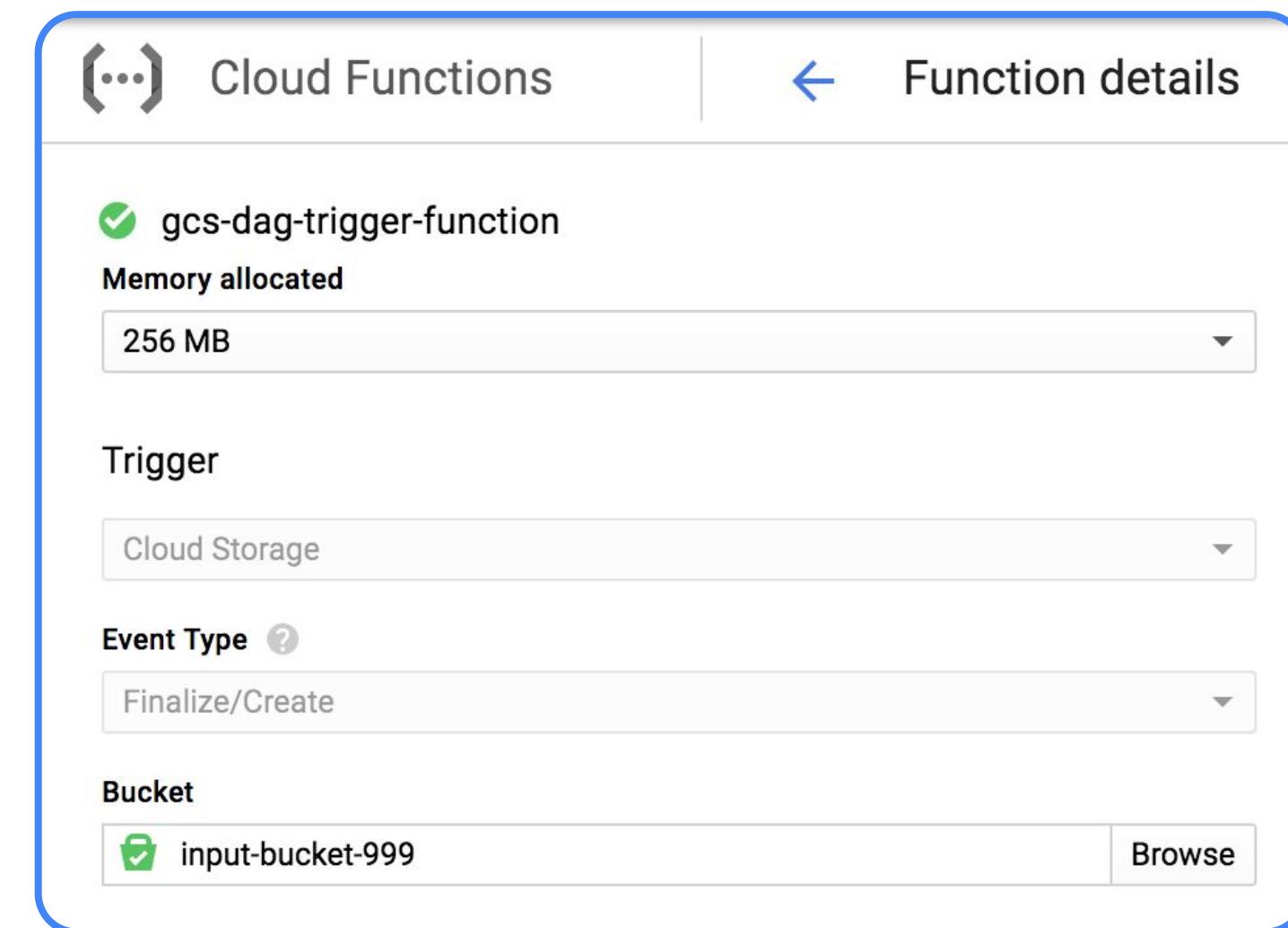
Use Cloud Functions to create an event-based push architecture



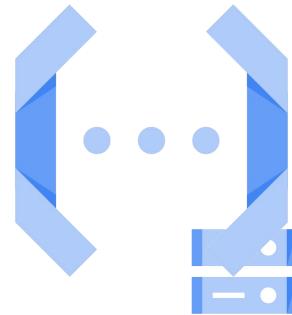
Example: Trigger an event when new data is loaded to Cloud Storage



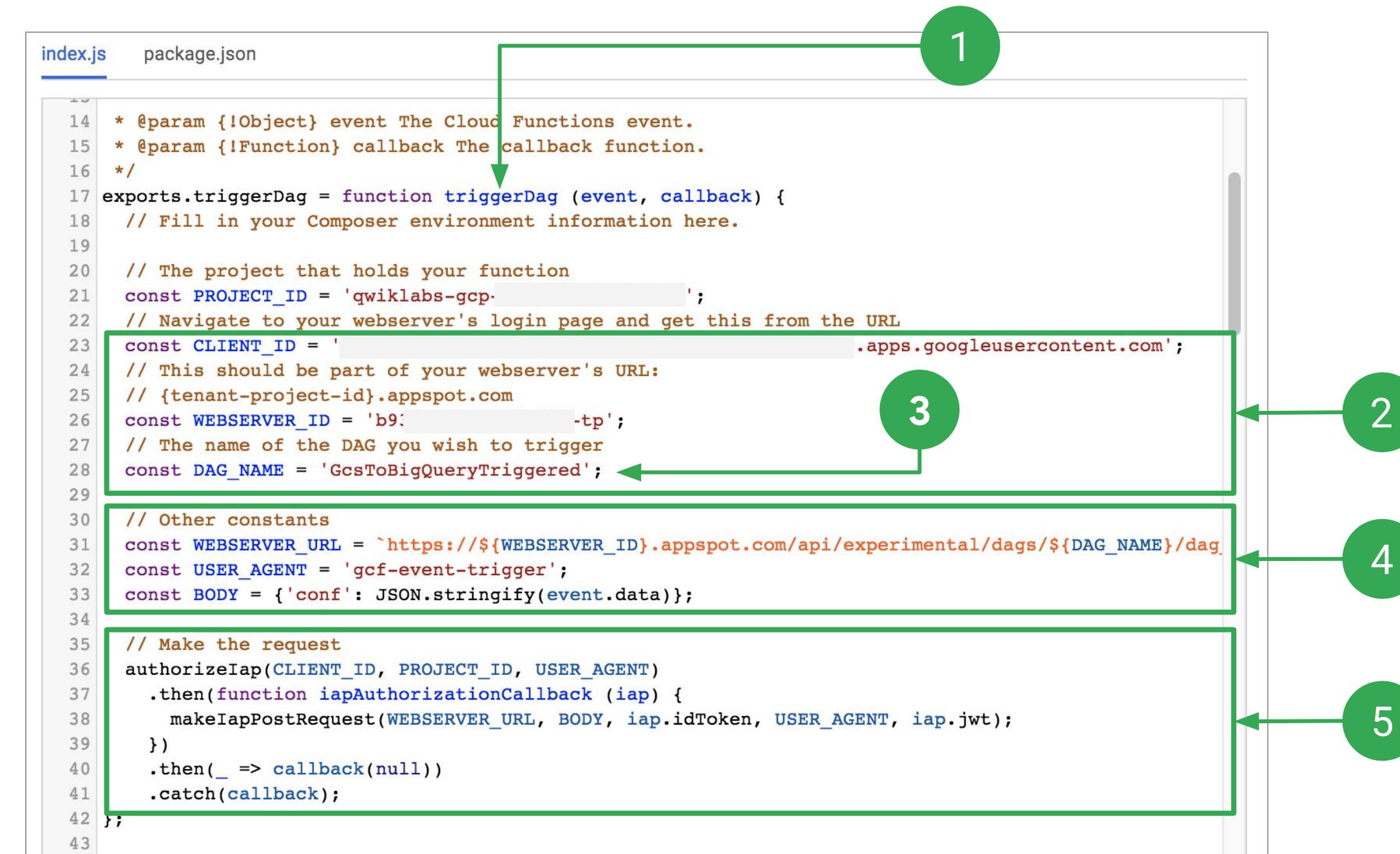
Creating a Cloud Function to trigger an Airflow DAG



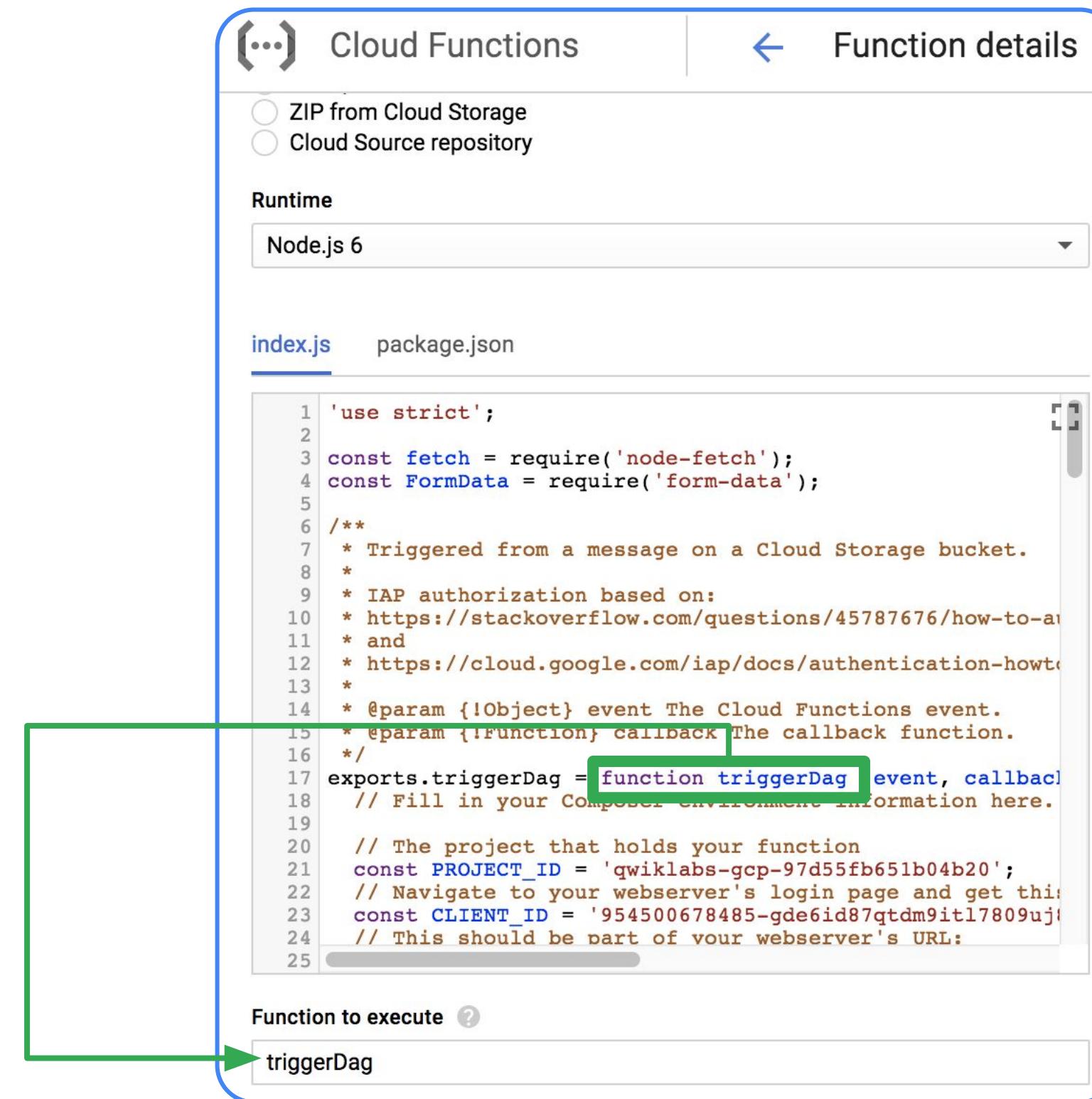
Creating a Cloud Function Cloud Storage trigger an Airflow DAG



Cloud Functions



Be sure to specify the function you want Cloud Functions to call in your script

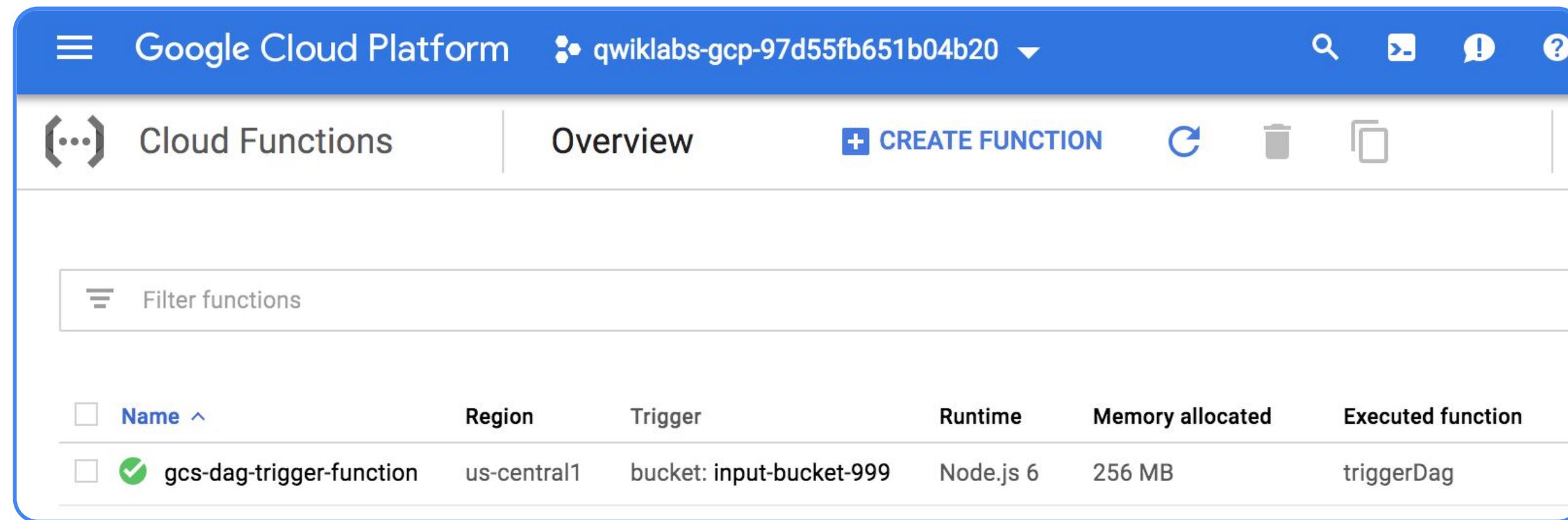


The screenshot shows the 'Function details' page for a Cloud Function. The runtime is set to Node.js 6. The code editor displays 'index.js' with the following content:

```
1 'use strict';
2
3 const fetch = require('node-fetch');
4 const FormData = require('form-data');
5
6 /**
7 * Triggered from a message on a Cloud Storage bucket.
8 *
9 * IAP authorization based on:
10 * https://stackoverflow.com/questions/45787676/how-to-authenticate-a-cloud-storage-trigger-without-a-service-account-key
11 * and
12 * https://cloud.google.com/iap/docs/authentication-howto
13 *
14 * @param {Object} event The Cloud Functions event.
15 * @param {Function} callback The callback function.
16 */
17 exports.triggerDag = function triggerDag(event, callback) {
18   // Fill in your Composer environment information here.
19
20   // The project that holds your function
21   const PROJECT_ID = 'qwiklabs-gcp-97d55fb651b04b20';
22   // Navigate to your webserver's login page and get this
23   const CLIENT_ID = '954500678485-gde6id87qtdm9itl7809uj';
24   // This should be part of your webserver's URL:
25 }
```

A green arrow points from the 'triggerDag' line in the code editor to the 'Function to execute' dropdown at the bottom of the page, which also contains the value 'triggerDag'.

Cloud Function is triggered whenever a new file is loaded to a specific Cloud Storage bucket



The screenshot shows the Google Cloud Platform Cloud Functions Overview page. The top navigation bar includes the Google Cloud Platform logo, the project name "qwiklabs-gcp-97d55fb651b04b20", and various navigation icons. Below the navigation bar, there are tabs for "Cloud Functions" and "Overview", with "Cloud Functions" selected. A "CREATE FUNCTION" button is visible. The main area displays a table of functions. The table has columns: Name, Region, Trigger, Runtime, Memory allocated, and Executed function. One function is listed: "gcs-dag-trigger-function" in the "us-central1" region, triggered by "bucket: input-bucket-999", using "Node.js 6" runtime, with 256 MB memory allocated, and the executed function being "triggerDag". A "Filter functions" input field is also present.

Name	Region	Trigger	Runtime	Memory allocated	Executed function
gcs-dag-trigger-function	us-central1	bucket: input-bucket-999	Node.js 6	256 MB	triggerDag

Monitor current and historical workflow progress

The screenshot shows the Airflow web interface with the following details:

- Header:** Airflow logo, DAGs, Data Profiling, Browse, Admin, Docs, About, 20:52 UTC, and a power icon.
- Section Title:** Dag Runs
- Actions:** List (273), Create, Add Filter, With selected, and a Search input field.
- Filter Bar:** A dropdown filter for "Dag Id" set to "equals" with the value "composer_sample_simple_greet". There is also a "Reset Filters" button.
- Table:** A list of Dag Runs with the following columns: State, Dag Id, Execution Date, Run Id, and External Trigger. The table contains six rows, each corresponding to a successful run of the "composer_sample_simple_greeting" DAG on different dates from 2018-09-26T00:00:00 to 2018-09-30T00:00:00.

	State	Dag Id	Execution Date	Run Id	External Trigger	
<input type="checkbox"/>		success	composer_sample_simple_greeting	09-30T00:00:00	scheduled_2018-09-30T00:00:00	
<input type="checkbox"/>		success	composer_sample_simple_greeting	09-29T00:00:00	scheduled_2018-09-29T00:00:00	
<input type="checkbox"/>		success	composer_sample_simple_greeting	09-28T00:00:00	scheduled_2018-09-28T00:00:00	
<input type="checkbox"/>		success	composer_sample_simple_greeting	09-27T00:00:00	scheduled_2018-09-27T00:00:00	
<input type="checkbox"/>		success	composer_sample_simple_greeting	09-26T00:00:00	scheduled_2018-09-26T00:00:00	

Quickly gauge pipeline health of DAG runs

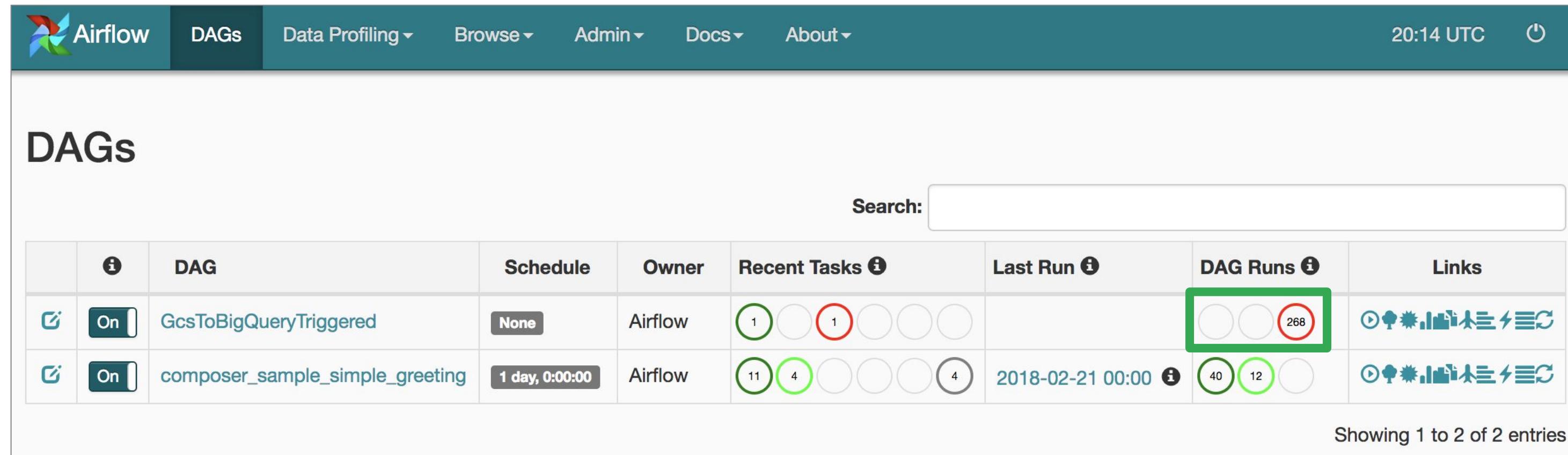
The screenshot shows the Airflow web interface with the 'DAGs' tab selected. The top navigation bar includes links for 'Data Profiling', 'Browse', 'Admin', 'Docs', and 'About'. The timestamp '20:14 UTC' and a power icon are also present. The main content area is titled 'DAGs' and features a search bar. Below it is a table with the following data:

	i DAG	Schedule	Owner	Recent Tasks i	Last Run i	DAG Runs i	Links
	On GcsToBigQueryTriggered	None	Airflow	1 1 1 1 1		268	
	On composer_sample_simple_greeting	1 day, 0:00:00	Airflow	11 4 4 4 4	2018-02-21 00:00 i	40 12 1	

At the bottom right of the table, it says 'Showing 1 to 2 of 2 entries'.

Which pipeline is healthier?

Quickly gauge pipeline health of DAG runs



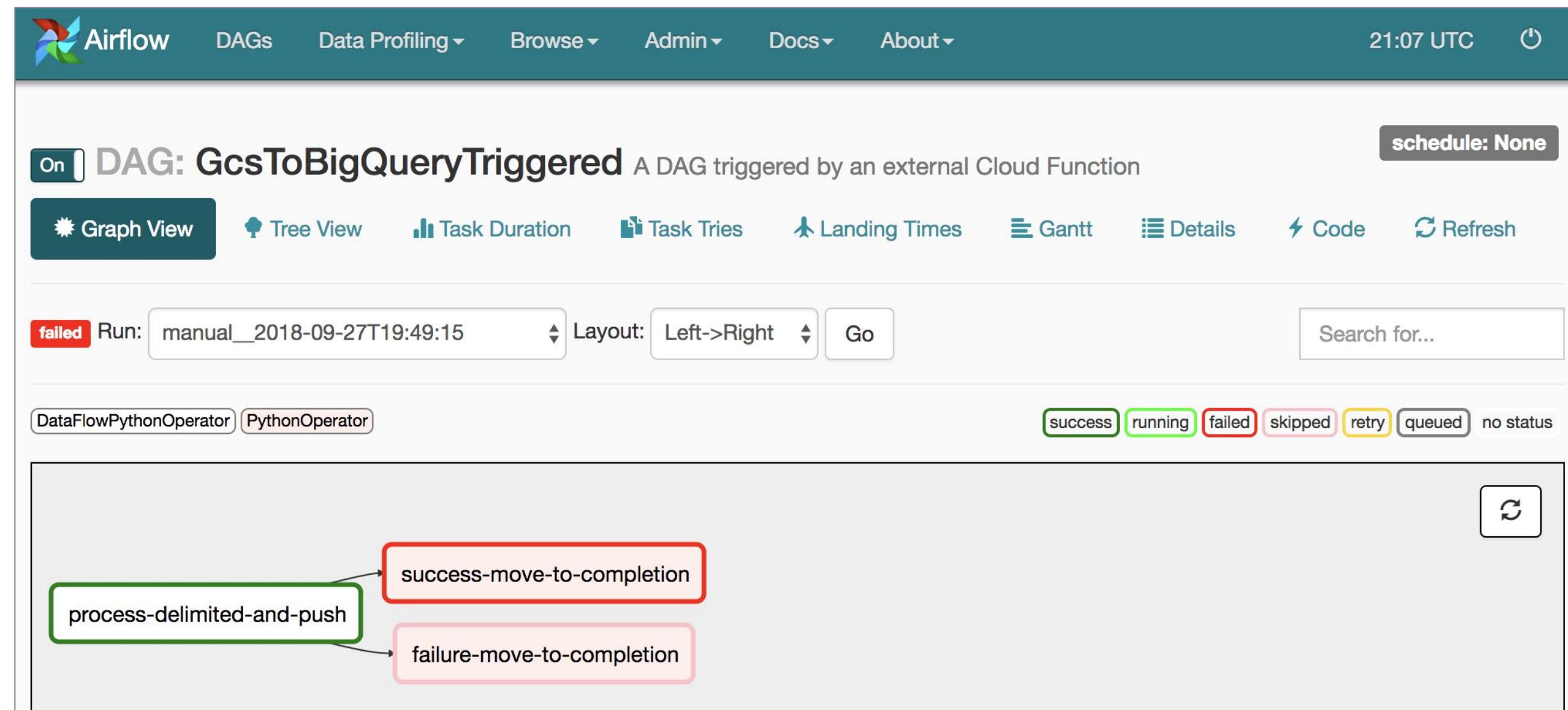
The screenshot shows the Airflow web interface for managing Data Pipelines (DAGs). The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, and About, along with the current time (20:14 UTC) and a power icon.

The main content area is titled "DAGs" and features a search bar labeled "Search:". Below the search bar is a table listing two DAGs:

	i	DAG	Schedule	Owner	Recent Tasks <i>i</i>	Last Run <i>i</i>	DAG Runs <i>i</i>	Links
		GcsToBigQueryTriggered	None	Airflow				
		composer_sample_simple_greeting	1 day, 0:00:00	Airflow		2018-02-21 00:00		

At the bottom of the table, it says "Showing 1 to 2 of 2 entries".

Quickly gauge pipeline health of DAG runs



Monitor and troubleshoot Airflow step errors in logs

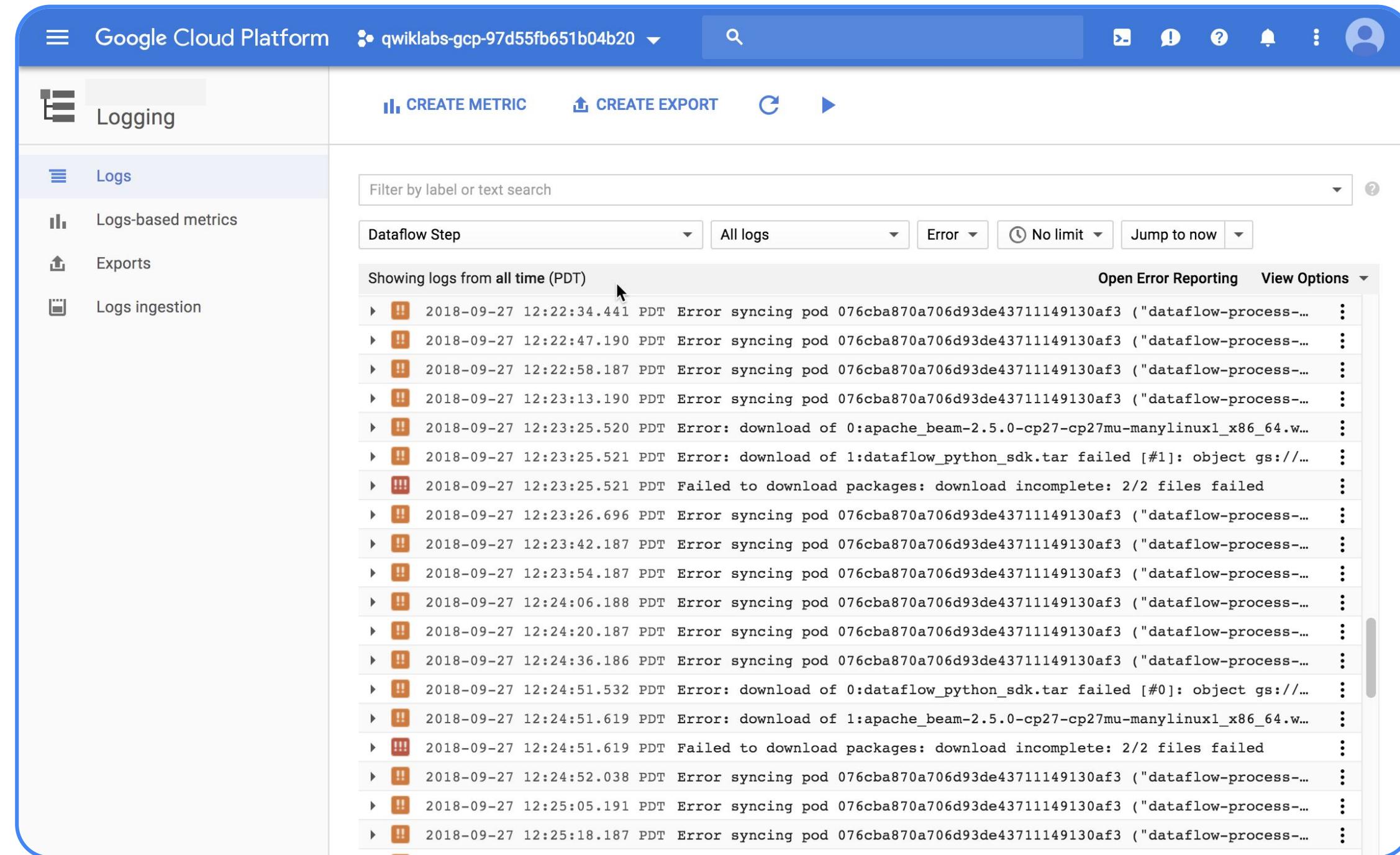
The screenshot shows the Airflow web interface for a DAG named "GcsToBigQueryTriggered". The task instance is labeled "success-move-to-completion" and was run on 2018-09-27 at 19:49:15. The "Log" tab is selected, displaying the following log output:

```
*** Reading remote log from gs://us-central1-evan-composer-0e85530c-bucket/logs/GcsToBigQueryTriggered/success-move-to-complet
[2018-09-27 20:03:38,633] {cli.py:374} INFO - Running on host airflow-worker-7bcfcc477b-mdgv7
[2018-09-27 20:03:38,698] {models.py:1196} INFO - Dependencies all met for <TaskInstance: GcsToBigQueryTriggered.success-move-
[2018-09-27 20:03:38,729] {models.py:1196} INFO - Dependencies all met for <TaskInstance: GcsToBigQueryTriggered.success-move-
[2018-09-27 20:03:38,729] {models.py:1406} INFO -
Starting attempt 1 of

[2018-09-27 20:03:38,751] {models.py:1427} INFO - Executing <Task(PythonOperator): success-move-to-completion> on 2018-09-27 20:03:38,751
[2018-09-27 20:03:38,751] {base_task_runner.py:115} INFO - Running: ['bash', '-c', u'airflow run GcsToBigQueryTriggered success-
[2018-09-27 20:03:40,439] {base_task_runner.py:98} INFO - Subtask: [2018-09-27 20:03:40,439] {__init__.py:45} INFO - Using execu
```

errors.HttpError: <HttpError 400 when requesting
https://www.googleapis.com/storage/v1/b/input-bucket-999/o/usa_names.csv/copyTo/b/gs%3A%2F%2Foutput-bucket-999%2F/o/success%2F2018-09-27%2Fusa_names.csv?alt=json returned "Invalid bucket name: 'gs://output-bucket-999/'>

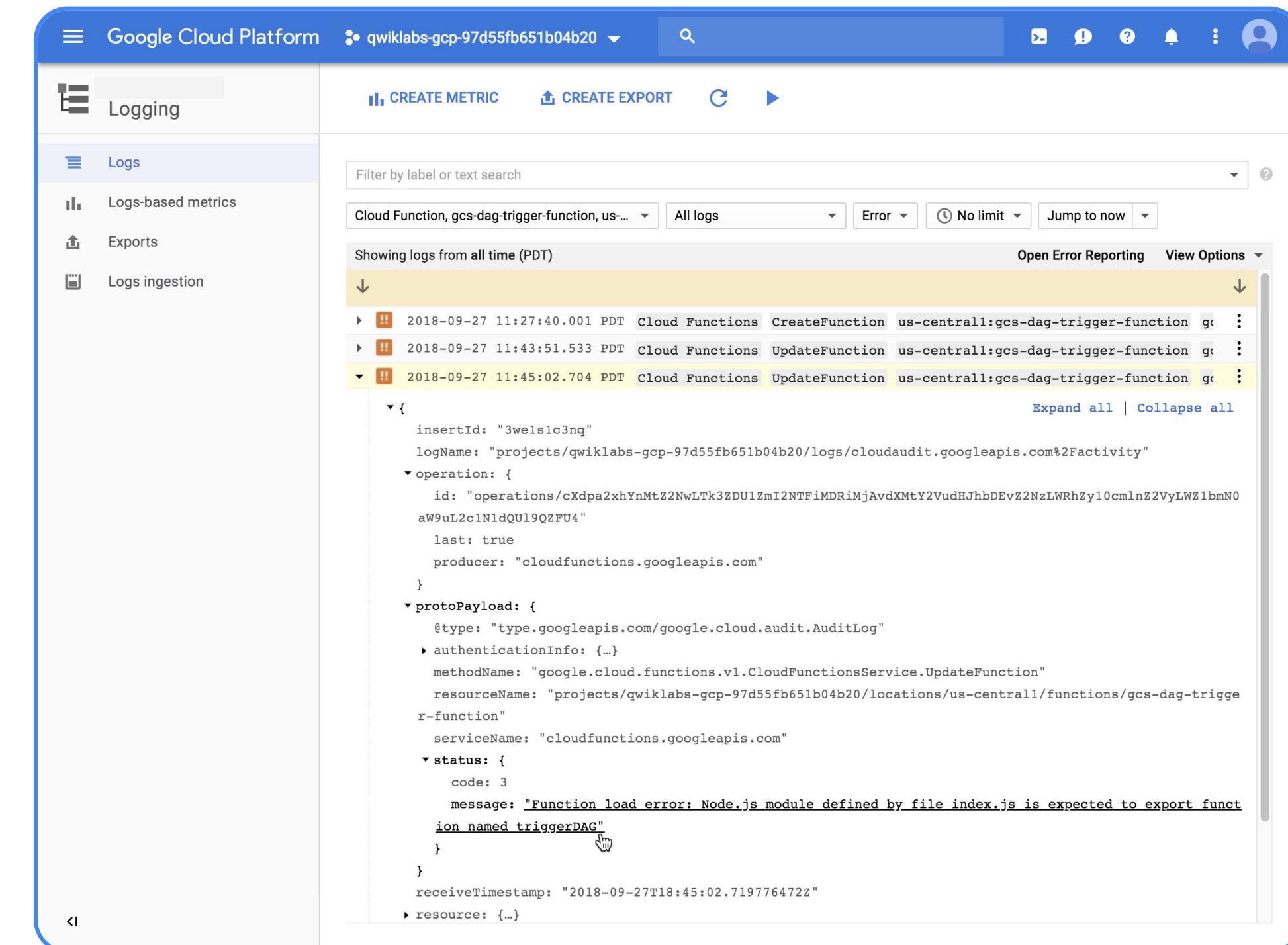
Monitor Dataflow health in Cloud Logging



The screenshot shows the Google Cloud Platform Logging interface. The left sidebar has 'Logs' selected. The main area shows a list of log entries for a 'Dataflow Step'. The logs are filtered to show 'Error' type messages from the 'All logs' category. The list includes numerous entries from September 27, 2018, at various times, all reporting errors related to syncing pods and package downloads.

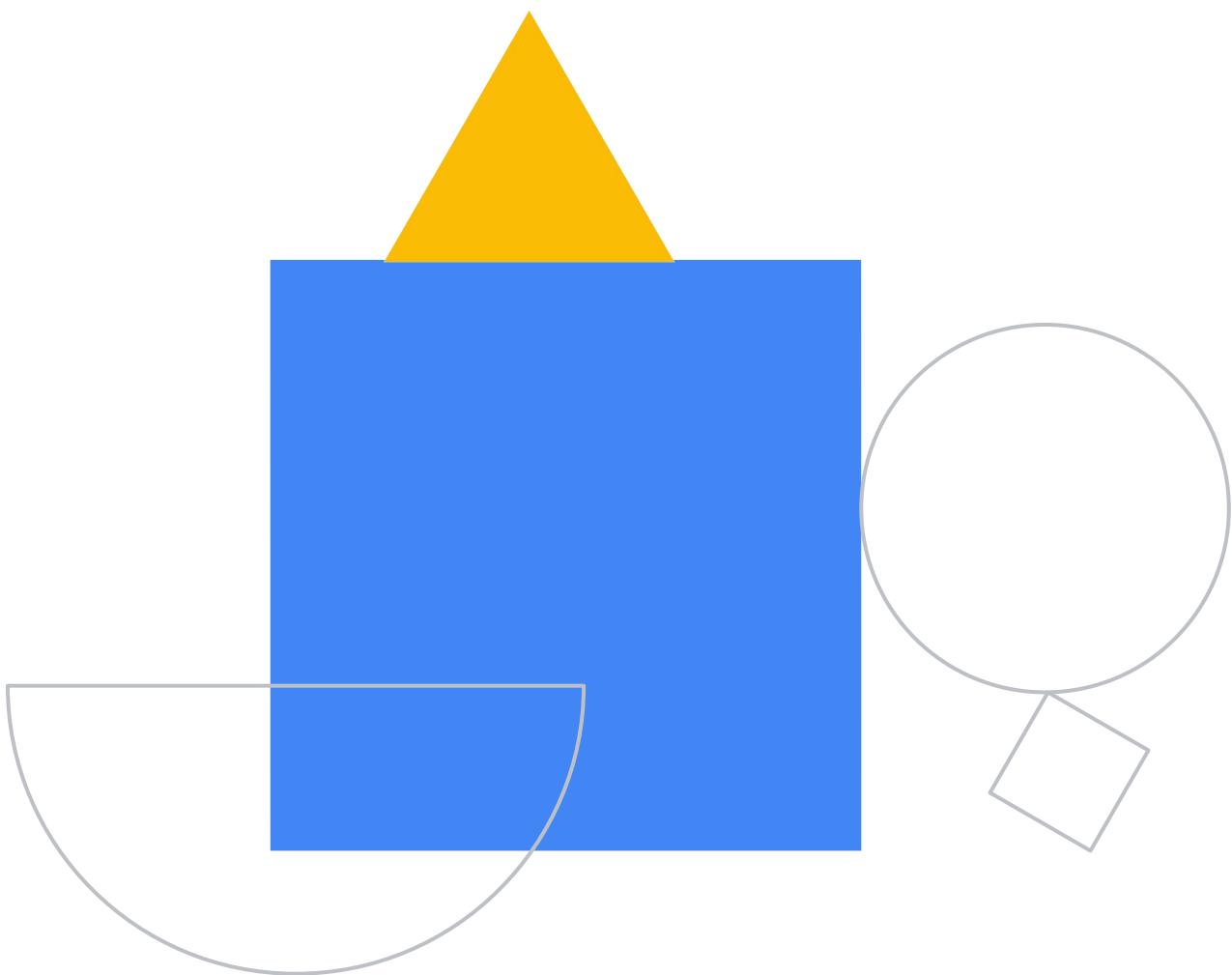
Date	Time	Error Description
2018-09-27	12:22:34.441 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:22:47.190 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:22:58.187 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:23:13.190 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:23:25.520 PDT	Error: download of 0:apache_beam-2.5.0-cp27-cp27mu-manylinux1_x86_64.w...
2018-09-27	12:23:25.521 PDT	Error: download of 1:dataflow_python_sdk.tar failed [#1]: object gs://...
2018-09-27	12:23:25.521 PDT	Failed to download packages: download incomplete: 2/2 files failed
2018-09-27	12:23:26.696 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:23:42.187 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:23:54.187 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:24:06.188 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:24:20.187 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:24:36.186 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:24:51.532 PDT	Error: download of 0:dataflow_python_sdk.tar failed [#0]: object gs://...
2018-09-27	12:24:51.619 PDT	Error: download of 1:apache_beam-2.5.0-cp27-cp27mu-manylinux1_x86_64.w...
2018-09-27	12:24:51.619 PDT	Failed to download packages: download incomplete: 2/2 files failed
2018-09-27	12:24:52.038 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:25:05.191 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:25:18.187 PDT	Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")

Monitor Cloud Function health in Cloud Logging



Lab Intro

An Introduction to Cloud Composer



Lab objectives

- 01 Use the Cloud Console to create a Cloud Composer environment
- 02 View and run a DAG in the Airflow web interface
- 03 View the results of the wordcount job in storage



Summary

Cloud Data Fusion for building data pipelines.

Cloud Composer, Cloud Functions and Cloud Scheduler
are the glue for your data pipelines.

