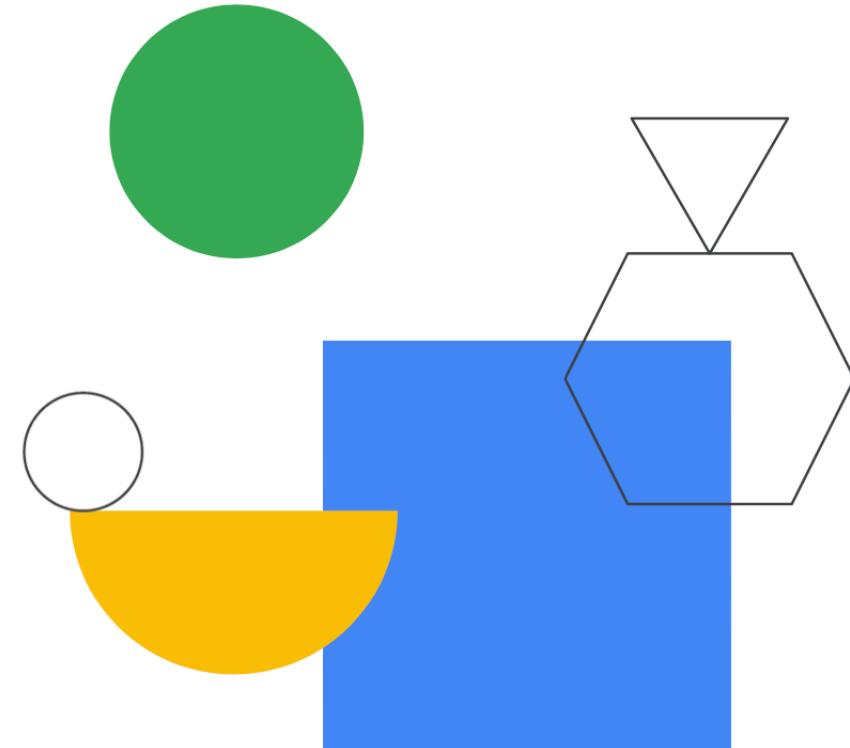
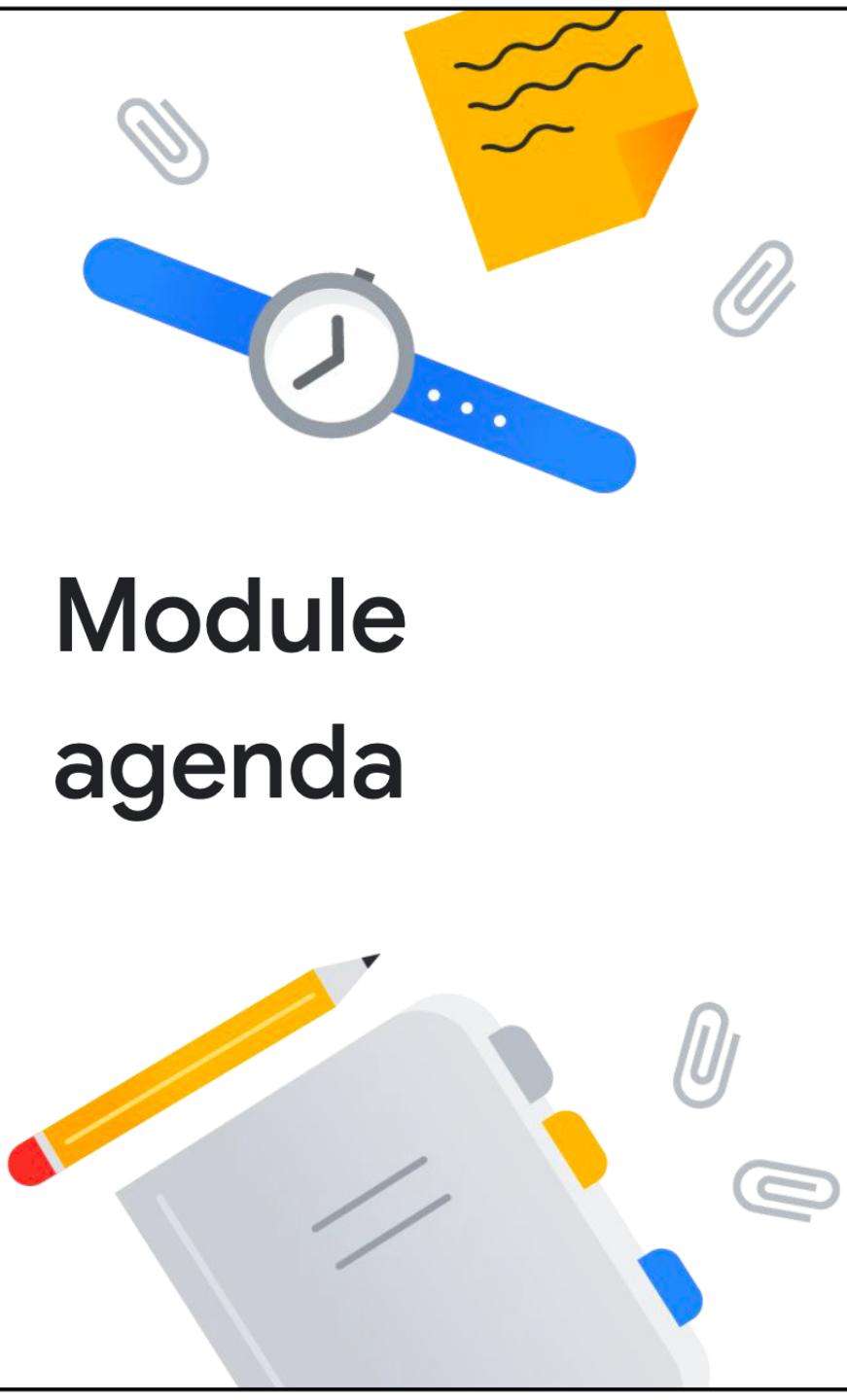


Manage Data Pipelines with Cloud Data Fusion and Cloud Composer



Module agenda



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



Building Batch Data Pipelines Visually with Cloud Data Fusion

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



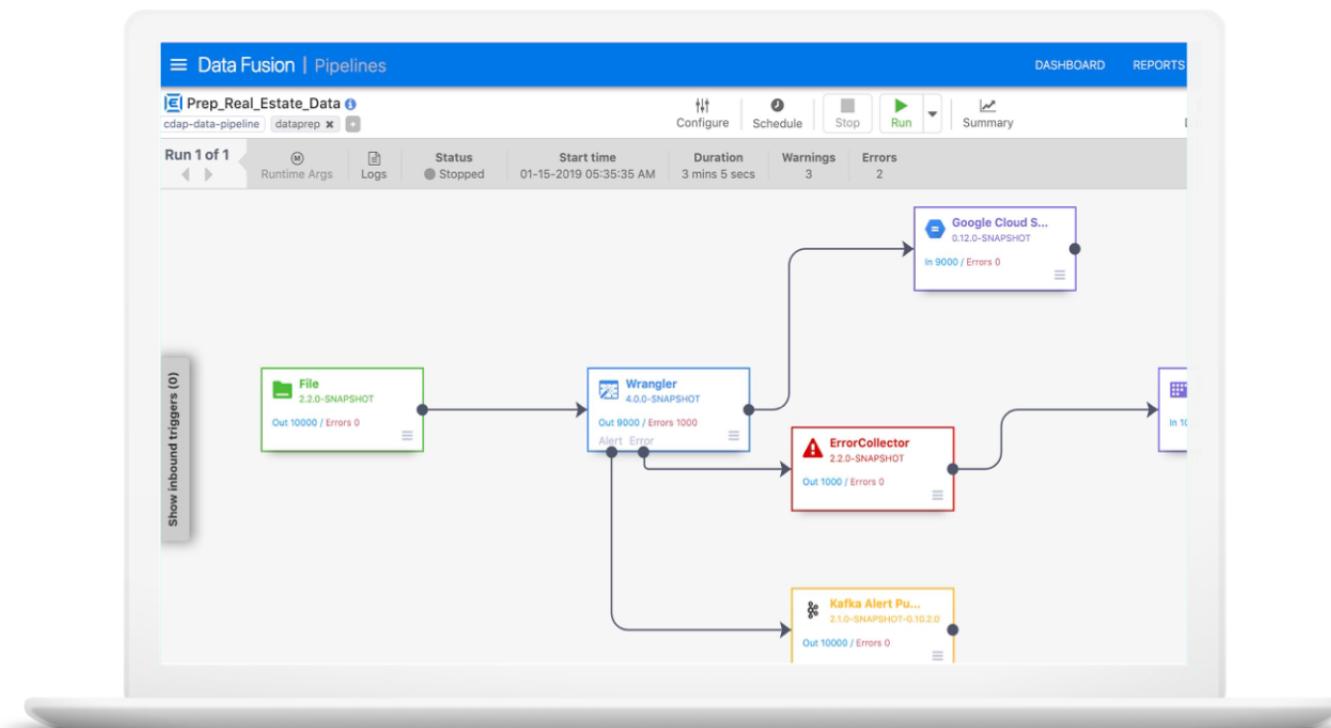
Increase flexibility

The screenshot shows the Google Data Fusion Preparation interface. At the top, there's a blue header bar with the title "Data Fusion | Preparation" and a "DASHBOARD" button. Below the header is a toolbar with various icons for file operations. The main area is a data editor titled "sales_clean.txt". It displays a table with 11 rows of house sales data. The columns are labeled: price (Double), city (String), zip (String), type (String), beds (String), baths (Double), size (Integer), lot_size (Long), and stories (Integer). The data shows various properties in Santa Clara and Palo Alto with different characteristics and prices.

	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

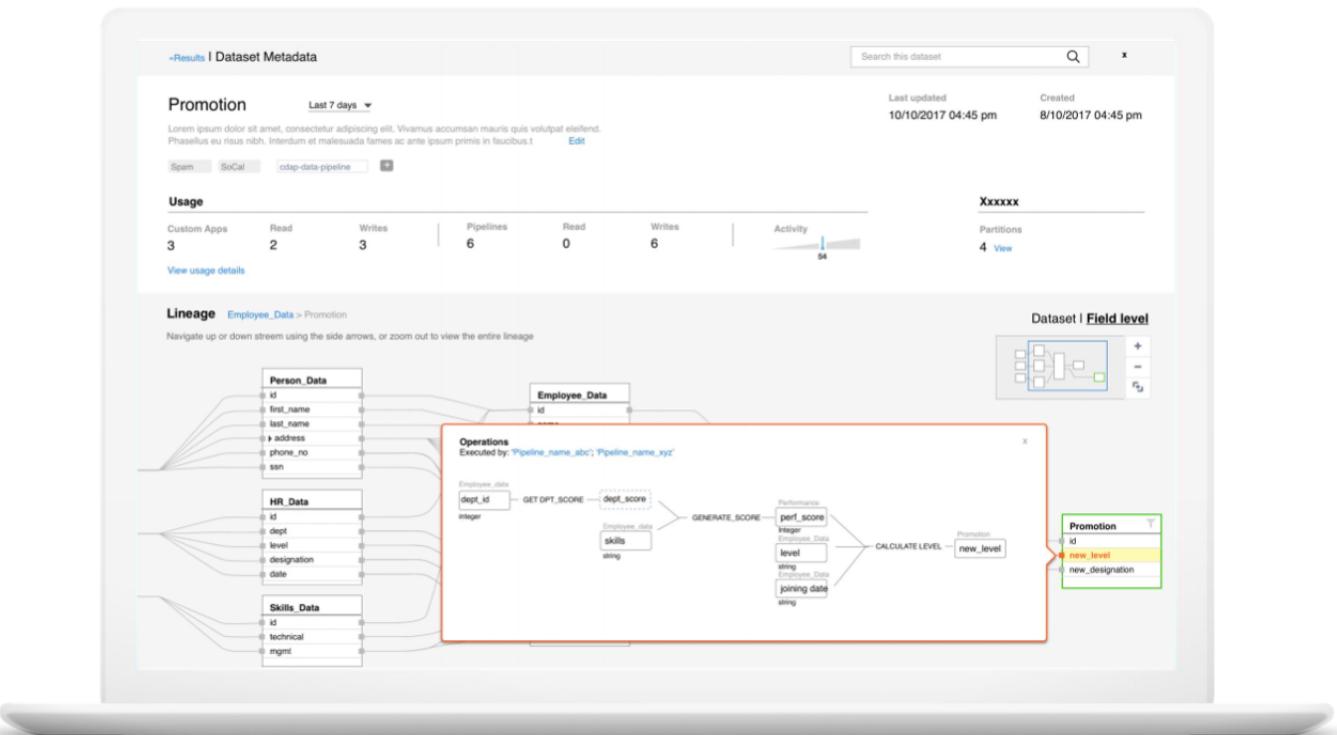
Tags and Properties support

- Pipeline
- Dataset
- Schema

Search integrated entities by

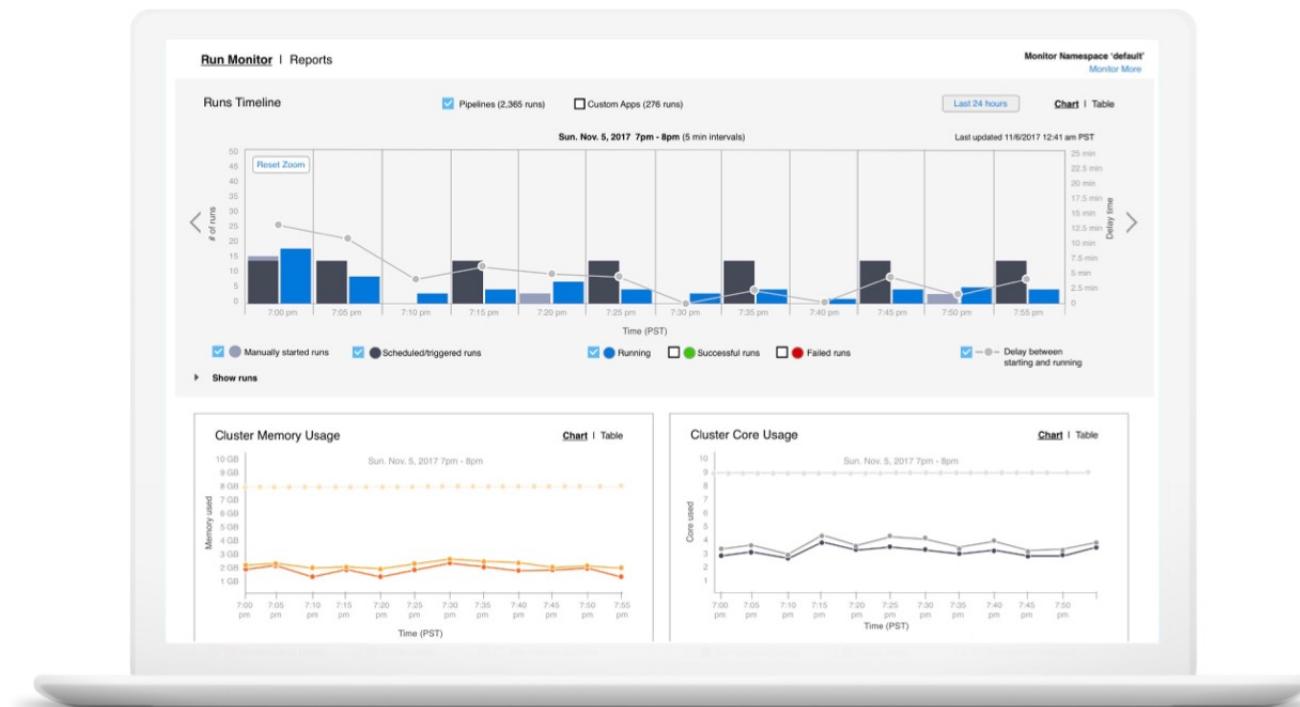
- Keyword
- Schema name and type

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

Wrangler

titanic.csv

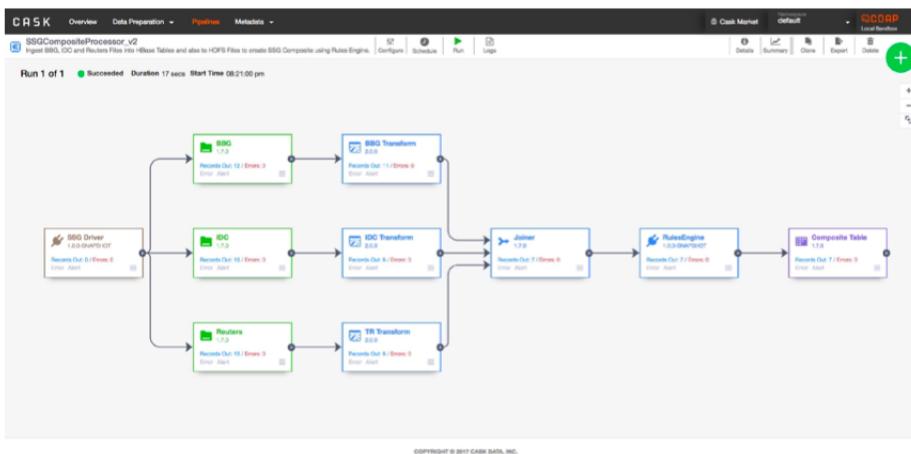
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	1	0	3	Brund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	2	1	1	Cumings, Mrs. John Bradley (Florence Brigg Thayer)	female	38	1	0	PC 17599
3	3	1	3	Heikkinen, Mrs. Lura	female	26	0	0	STON/OAK 314
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	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	35	0	0	330877
7	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	8	0	3	Paterson, Master. Gosta Leonard	male	2	3	1	349809
9	9	1	3	Johnson, Mrs. Oscar W (Elizabeth Vilhelmina Berg)	female	27	0	2	347742
10	10	1	2	Nasser, Mrs. Nicholas (Adele Achen)	female	14	1	0	237396
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	Saunderscock, Mr. William Henry	male	20	0	0	A/5 2151
14	14	0	3	Anderson, Mr. Anders John	male	39	1	0	347082
15	15	0	3	Weirum, Miss. Hulda Amanda Adolfs	female	14	0	0	350406
16	16	1	3	Heude, Mrs. (Man D'Kronos)	female	55	0	0	348706

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

- 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, a sidebar lists 'Rules Books (5)' and 'Rules (23)'. A search bar and a 'Create a New Rule' button are also present. The main area displays a rulebook named 'Client1CompositeRulebook' with the following details:

- Owner: rao
- Last Updated: 08-04-17 10:42
- Description: This defines the rulebook for the client composite dataset generation for 390
- Rules (9):
 - missing-age: If age is missing, send it to error
 - MyOverRule: Custom desc
 - coupon_req: coupon_freq either comes from BBG or Reuters
 - user_sector2: user_sector2 either comes from IDC or BBG
 - user_sector1: user_sector1 either comes from IDC or BBG
 - asset_id: For investment type is EQ or != set asset_id either from bbq or from idc
 - eps: EPS set either from BBG or Reuters
 - asset_id_type: For investment type is EQ or != - set asset_id type either from bbq or from idc
 - remove-fare-less-than-8.06: Send to errr fares that are less than 8.06

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.

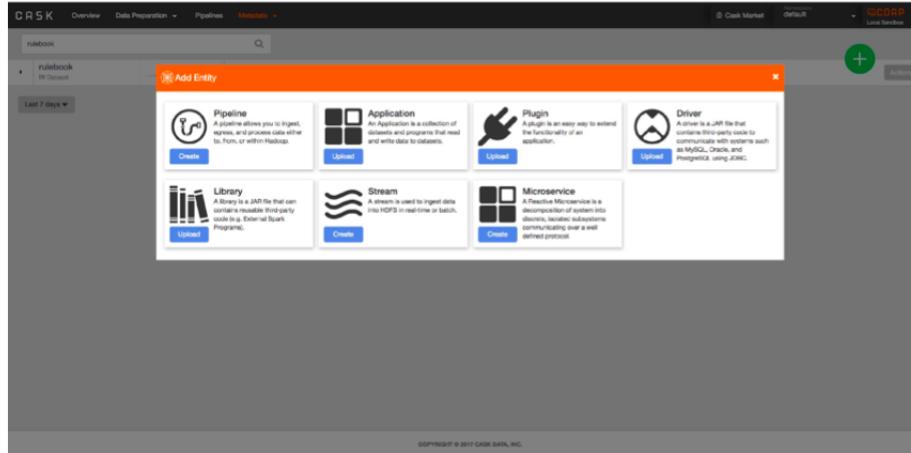
The screenshot shows the Cask Metadata Aggregator interface. At the top, there are tabs for Overview, Data Preparation, Pipelines, and Metadata. Below that, a sidebar lists 'rulebook' and 'rules'. A search bar and a 'Lineage' tab are also present. The main area displays a lineage diagram with nodes for 'para' (service), 'rulebook' (Table), and 'rules' (Table). Arrows indicate the flow of data from 'para' to both 'rulebook' and '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.

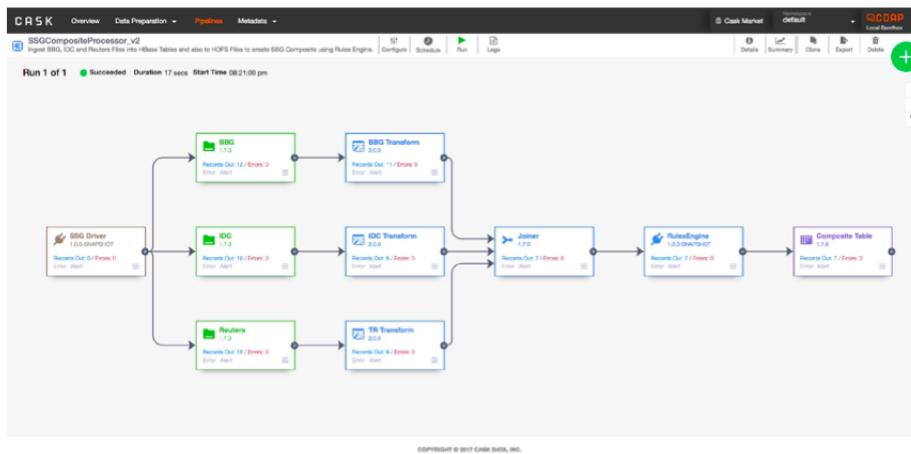
Components of Cloud Data Fusion



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

The screenshot shows the Cloud Data Fusion UI with the title "Cloud Data Fusion | Pipelines". The left sidebar includes sections for Control Center, Pipelines (selected), Studio, Transform, and Metadata. The main area displays a table of pipelines with columns for Type, Status, Last start time, Next run in, Total runs, and Tags. The table lists several entries, including a failed batch pipeline, running and succeeded batch pipelines, and a deployed realtime pipeline.

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"

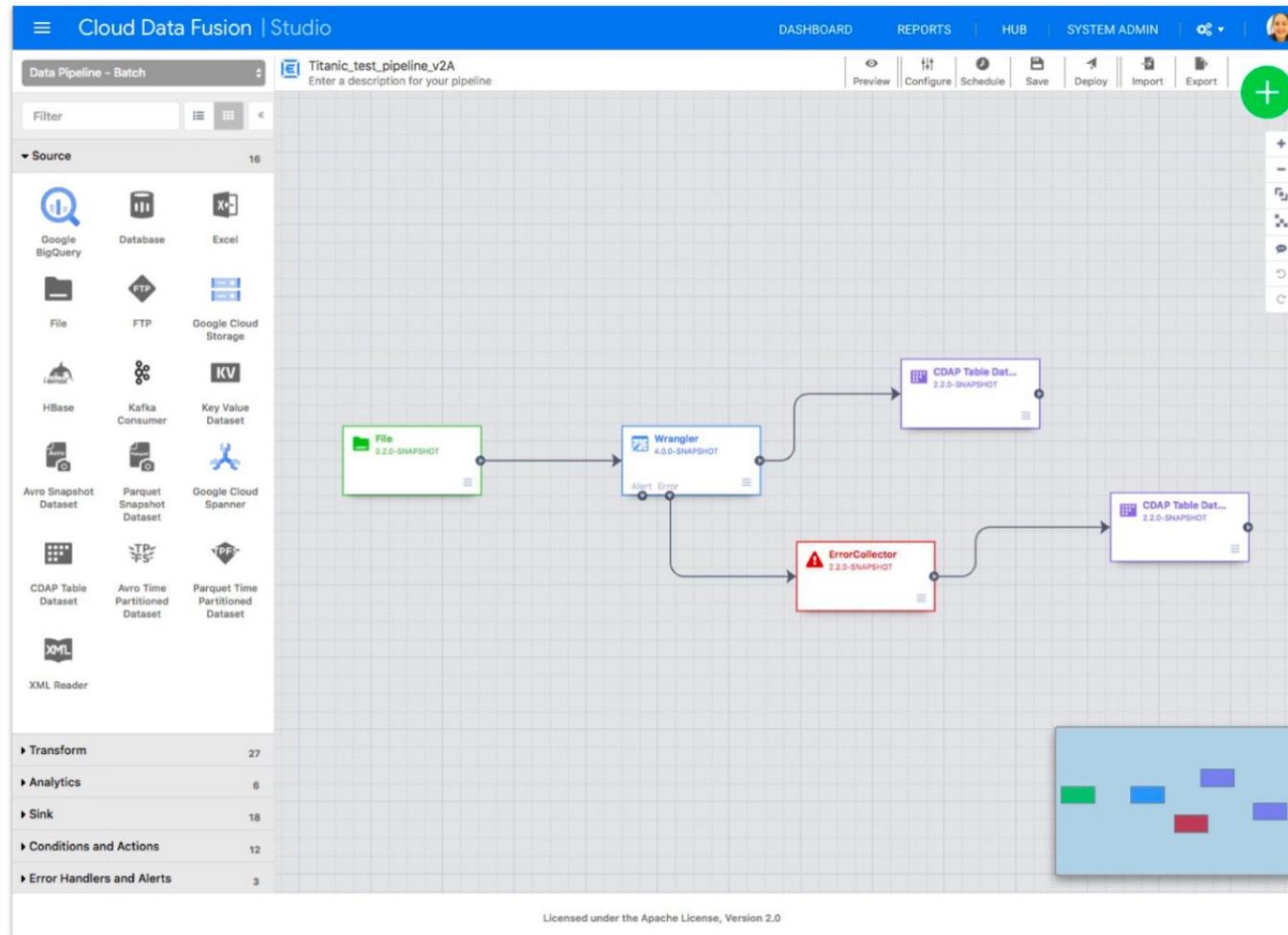
Displaying Applications, Datasets, sorted by Newest

26 entities

Entity Type	Name	Version	Programs	Operations	Writes	Actions
Data Pipeline	join-w-customer	1.0.0-SNAPSHOT	2	0	0	
Dataset	sales-with-zip		1	10	3	
Dataset	customers.csv		1	0	0	
Dataset	sales.csv		1	0	0	
Data Pipeline	sales-ingest	1.0.0-SNAPSHOT	2	0	0	
Dataset	sales		2	15	10	
Data Pipeline	customers-ingest	1.0.0-SNAPSHOT	2	0	0	
Dataset	customers		2	12	9	
Dataset	U.S_Chronic_Disease_Indicator...		1	0	0	
Data Pipeline	Mask_data	1.0.0-SNAPSHOT	1	42,119,097	42,119,097	
Application	ModelManagementApp	1.0.0-SNAPSHOT	1	0	0	
Dataset	experiment_model_meta		1	0	0	
Dataset	experiment_model_components		1	0	0	
Dataset	experiment_meta		1	1	0	
Data Pipeline	Titanic_test_pipeline_v1_test	1.0.0-SNAPSHOT	2	0	0	
Dataset	titanic.csv		0	0	0	
Data Pipeline	Titanic_test_pipeline_v1_test	1.0.0-SNAPSHOT	2	0	0	
Dataset	Titanic_test		0	0	0	
Dataset	Error_sink_titanic		0	0	0	
Application	dataprep	1.0.0-SNAPSHOT	1	1	0	
Dataset	connections		1	0	0	
Dataset	dataprepfs		1	0	0	
Dataset	workspace		1	2,499	239	

Pipelines

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



Wrangler

- Connections
- Transforms
- Data Quality
- Insights
- Functions

Cloud Data Fusion | Wrangler

titanic.csv Google Cloud Storage titanic.csv

Data Insights

Create a Pipeline More +

Columns (16) Transformation steps (20)

Search Column names ▾

	Integer PassengerId	String Survived	String Pclass	String Sex	Integer Age	String SibSp	String Parch	String Ticket	Double Fare	String Cabin	String 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

Integration metadata

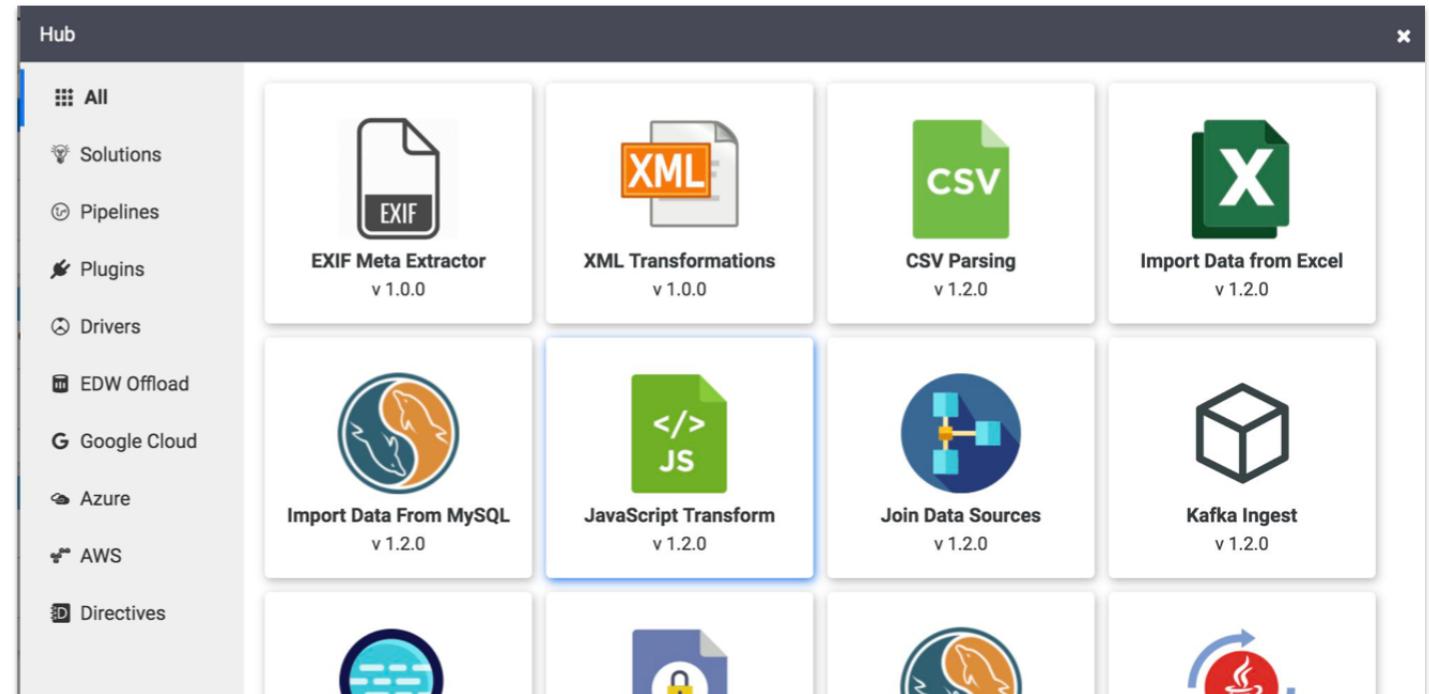
- Search
- Tags and Properties
- Lineage - Field and Data

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

Count	Dataset Name	Description
18	recipes	Dataset Created: Oct 12, 2018 Recipe store.
2	workspace	Dataset Created: Oct 12, 2018 Dataprep workspace dataset
4	dataprefs	Dataset Created: Oct 12, 2018 Store DataPrep Index files
18	connections	Dataset Created: Oct 12, 2018 DataPrep connections store.
1	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



Add entity

Pipeline
A pipeline allows you to create, manage, and operate complex batch and real-time workflows intuitively.
Create **Import**

Application
An application is a collection of datasets and programs that read and write data to datasets.
Upload

Plugin
A plugin is an easy way to extend the functionality of an application.
Upload

Driver
A driver is a JAR file that contains third-party code to communicate with systems such as MySQL, Oracle, and PostgreSQL using JDBC.
Upload

Library
A library is a JAR file that can contain reusable third-party code (e.g. external Spark programs).
Upload

Directive
A directive is a data manipulation instruction that can be used to perform data cleansing, transformation and filtering.
Upload

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

Transactions

NumCommittingChangeSets	0
NumInProgressTransactions	0
NumInvalidTransactions	0
NumCommittedChangeSets	0

Management | Configuration

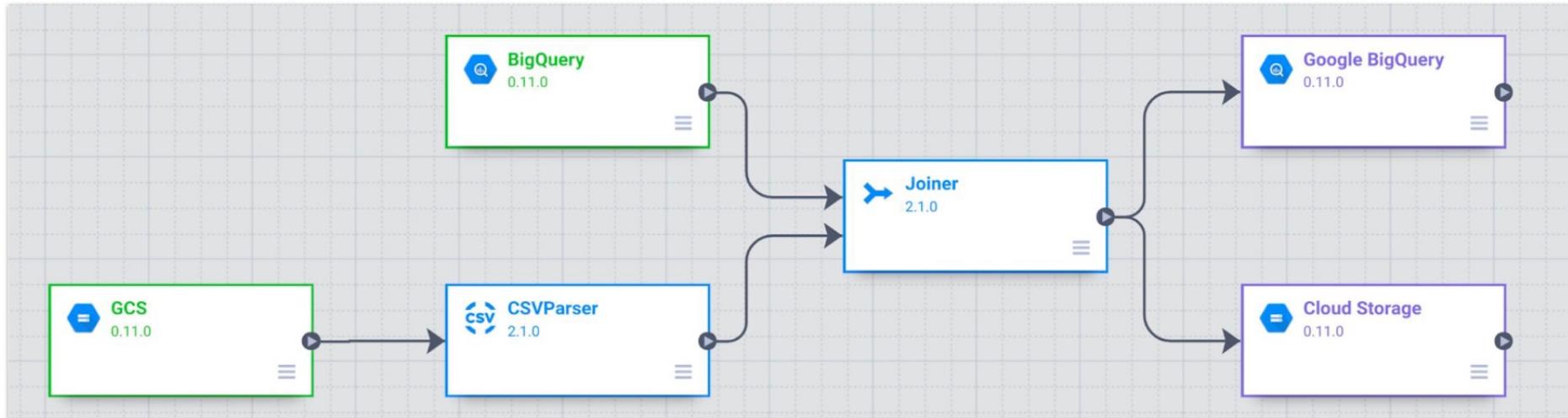
Namespaces (1)

System Compute Profiles (1)

Pipeline usage									
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)

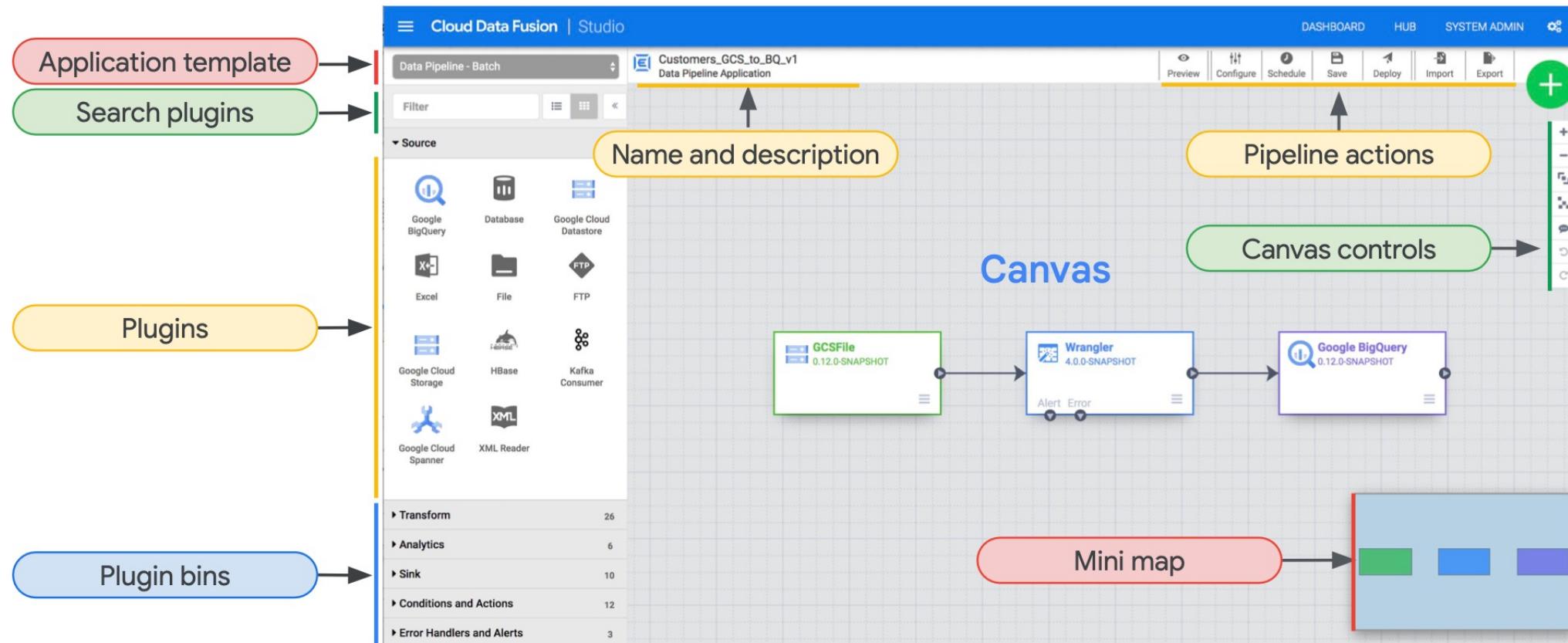


- Represented by a series of stages arranged in a DAG. This forms a one-way pipeline.
- Stages, which are the "nodes" in the pipeline graph, can be of different types.

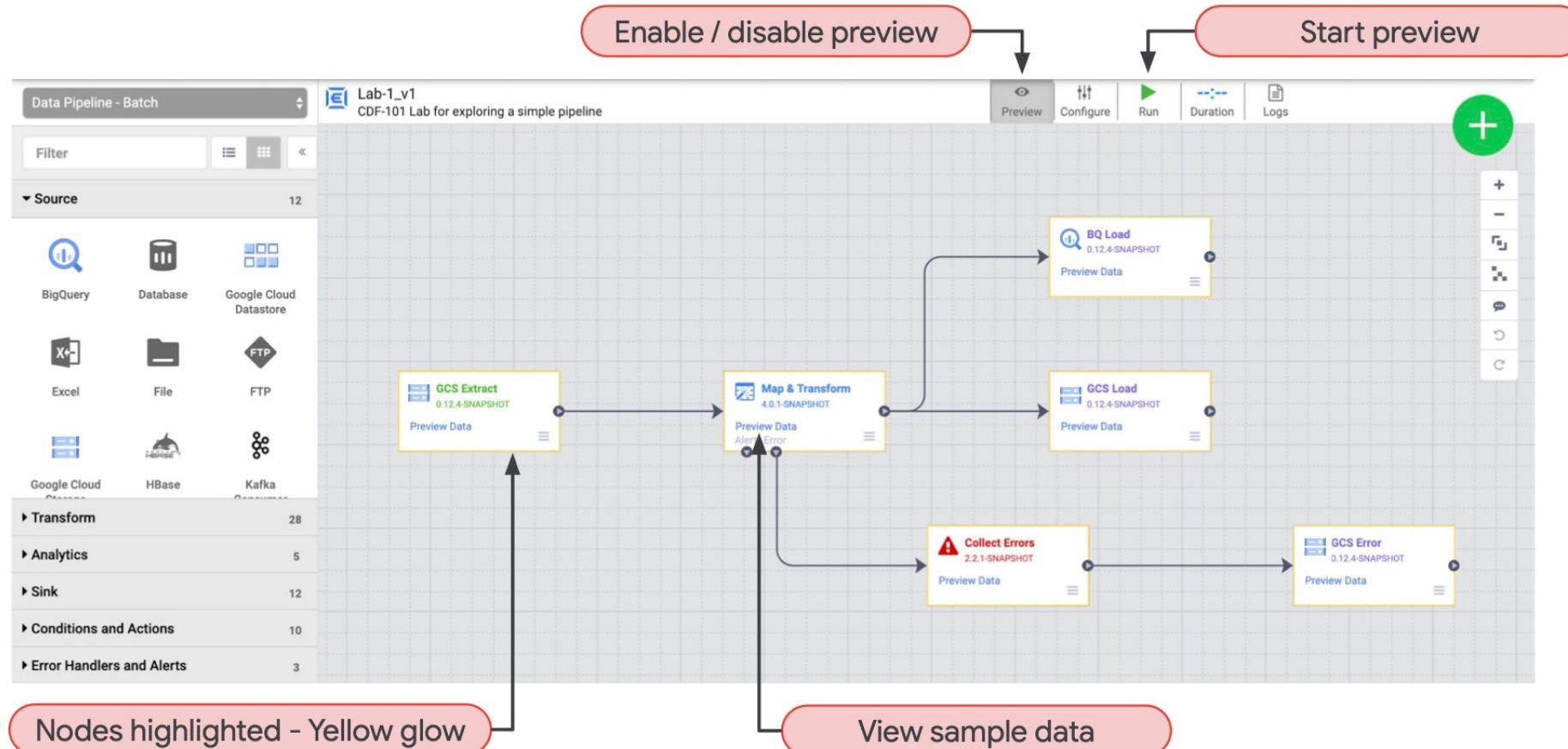
Data pipeline

- 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.

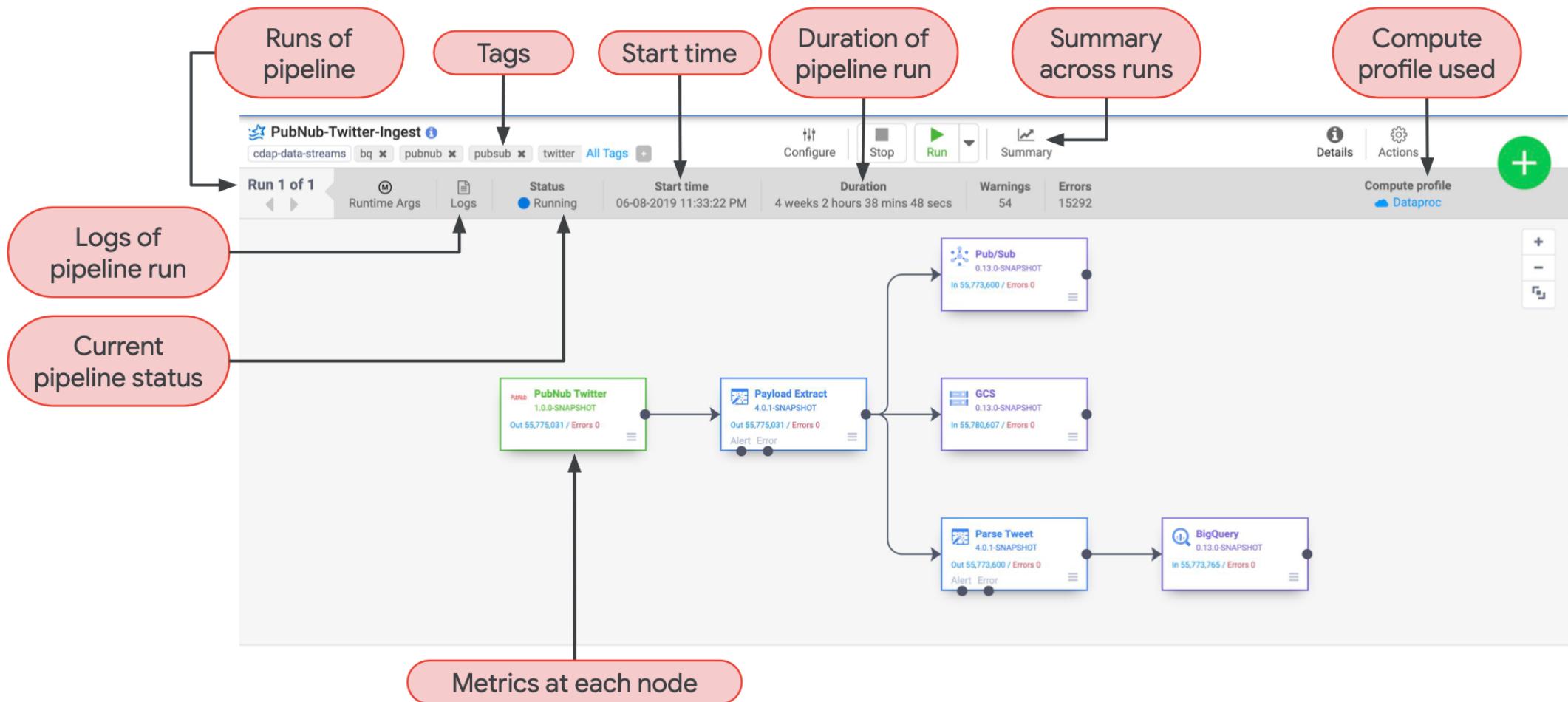
Studio is the UI where you create new pipelines



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

The screenshot shows the Cloud Data Fusion Pipeline interface. At the top, there's a blue header bar with the title "Cloud Data Fusion | Pipeline" and a sub-header "batch_pipeline". Below the header, there are tabs for "No Runs", "Runtime Args", "Logs", and "Status". On the right side of the header, there are buttons for "Configure", "Schedule", "Stop", "Run", and "Summary". To the right of these buttons are "Details" and "Actions" links, and a green circular button with a plus sign. The main content area is titled "Configure schedule for pipeline batch_pipeline". It has a "Basic" tab selected, showing settings for "Pipeline run repeats" (set to "Daily"), "Repeats every" (set to "1 day(s)"), and "Starting at" (set to "1:00 AM"). A "Summary" section below these fields states: "This pipeline is scheduled to run everyday, at 1:00AM. The pipeline cannot have concurrent runs." There are also fields for "Max concurrent runs" (set to "1") and "Compute profiles" (set to "Dataproc (Google Cloud Dataproc)"). At the bottom of the configuration window are two buttons: "Save and Start Schedule" (in blue) and "Save Schedule". On the far left and right edges of the main content area, there are grey vertical bars with text: "Show inbound triggers (0)" on the left and "Show outbound triggers (0)" on the right.

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

Relationship between fields of datasets: Provenance, Impact

☰ Cloud Data Fusion | Field Level Lineage

Enterprise Edition

doubleclick

« Back | Dataset

Field level lineage

Explore root cause and impact for each of the fields of the dataset

Cause for: doubleclick: campaign

1 Dataset

Dataset name	Field name
1 campaign	offset
	body

Dataset: doubleclick

8 Fields

Search by field name

Field name
advertiser_id
timestamp
campaign
referrer_url
campaign_id
landing_page_url
advertiser
landing_page_url_id

Last 7 days ▾

Impact for: doubleclick: campaign

1 Dataset

Dataset name	Field name
1 hits	campaign

Incoming operations Outgoing operations

```
graph TD; offset[campaign] --> campaign[campaign]; body[campaign] --> campaign; campaign --> referrer_url[referrer_url]; campaign --> campaign_id[campaign_id]; campaign --> landing_page_url[landing_page_url]; campaign --> advertiser[advertiser]; campaign --> landing_page_url_id[landing_page_url_id]
```

See every operation that is made on a field

Operations applied to a field

Cloud Data Fusion | Field Level Lineage

DASHBOARD HUB SYSTEM ADMIN Enterprise Edition

« Back | doubleclick
Dataset

Field level lineage

Cause operations for field 'campaign'

Operations between 'campaign' and 'doubleclick'

1 of 1

Last executed by '00-BigQuery_SQL_DoubleClick_v1' on 03-27-2019 09:48:28 AM

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 campaigns.c...	Unchanged as part of a join	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 code-free tool for data transformation. On the left, there's a sidebar with navigation options: Namespace (default), Control Center, Pipeline, List, Studio, **Wrangler** (which is selected and highlighted in blue), and Metadata. Below these are sections for Connections in "default" (Cloud SQL MySQL, TheSQL, Cloud SQL Postgresql, Kafka (0), S3 (1), Google Cloud Storage (2)), and a "+ Add Connection" button.

The main area is titled "Select data" and shows a pipeline flow. It starts with "credit_card_data.csv" from Google Cloud Storage, followed by "customers.csv". A "Create a Pipeline" button is located at the top right of this section. The pipeline then continues to "customers.csv" (which is also from Google Cloud Storage) and finally "employees.xml". Another "Create a Pipeline" button is located here as well.

On the right side of the interface, there's a detailed view of the "credit_card_data.csv" dataset. It shows a table with 8 rows and 6 columns. The columns are labeled: id, first_name, last_name, email, credit_card, and credit_card_type. The data includes various names and card details. Below the table, there are sections for "Columns (6)" and "Directives (4)". The directives listed are:

- # 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_type

At the bottom right, the text "Instance Id: cloud-data-fusion-demos/cdf-demo" is visible.

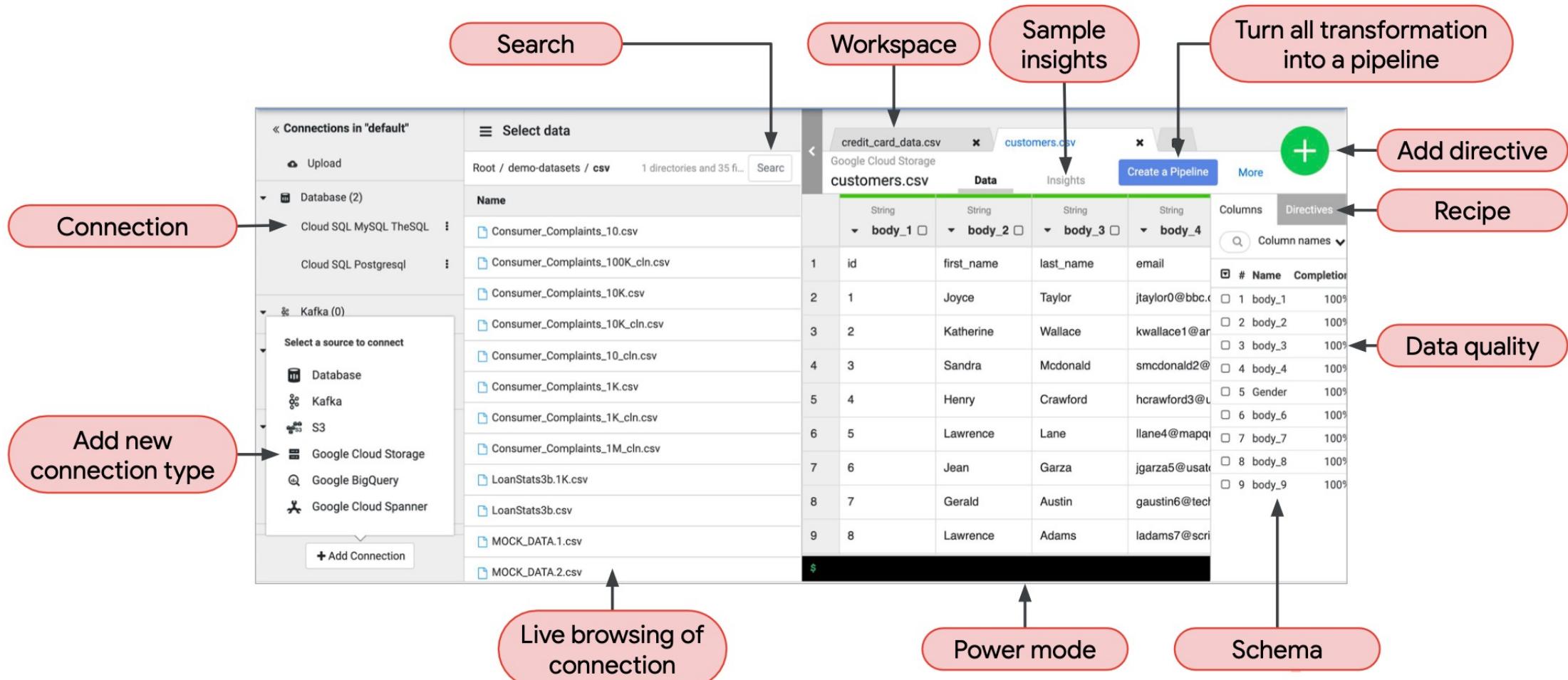
Wrangler UI overview for exploring datasets

The screenshot displays the Wrangler UI interface, which is a tool for exploring and transforming datasets. The interface is divided into several sections:

- Left Sidebar:** Shows "Connections in 'default'" with options for "Upload", "Database (2)" (Cloud SQL MySQL TheSQL, Cloud SQL Postgresql), "Kafka (0)", and "Select a source to connect" (Database, Kafka, S3, Google Cloud Storage, Google BigQuery, Google Cloud Spanner). A "Add Connection" button is at the bottom.
- Middle Left:** A "Select data" panel showing a list of CSV files in "Root / demo-datasets / csv". The list includes: Consumer_Complaints_10.csv, Consumer_Complaints_100K_cln.csv, Consumer_Complaints_10K.csv, Consumer_Complaints_10K_cln.csv, Consumer_Complaints_10_cln.csv, Consumer_Complaints_1K.csv, Consumer_Complaints_1K_cln.csv, Consumer_Complaints_1M_cln.csv, LoanStats3b.1K.csv, LoanStats3b.csv, MOCK_DATA.1.csv, and MOCK_DATA.2.csv.
- Right Side:** Two main data preview panels. The top one shows "credit_card_data.csv" and "customers.csv" from "Google Cloud Storage". The bottom one shows "customers.csv" with the following schema and data:

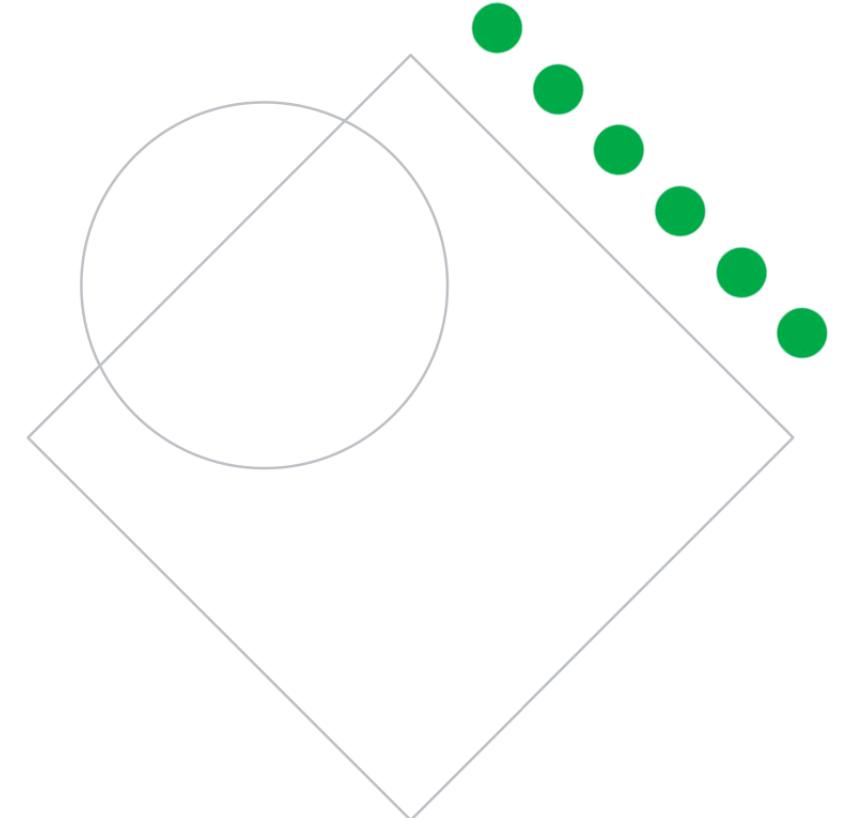
	String	String	String	String
	body_1	body_2	body_3	body_4
1	id	first_name	last_name	email
2	1	Joyce	Taylor	jTaylor0@bbc.co.uk
3	2	Katherine	Wallace	kwallace1@aristotele.com
4	3	Sandra	Mcdonald	smcdonald2@tutu.com
5	4	Henry	Crawford	hcrawford3@united.com
6	5	Lawrence	Lane	llane4@mapquest.com
7	6	Jean	Garza	jgarza5@usatoday.com
8	7	Gerald	Austin	gaustin6@techcrunch.com
9	8	Lawrence	Adams	ladams7@scripting.com
- Bottom Right:** A "Columns" section with a search bar and dropdown for "Column names". It also includes a "Directives" section with checkboxes for "# Name", "Completion", and "Transformation". Below these are checkboxes for columns 1 through 9, each with a completion percentage (e.g., body_1: 100%, body_2: 100%, ..., body_9: 100%).

Wrangler UI overview for exploring datasets



Lab Intro

Building and Executing a Pipeline
Graph in Cloud Data Fusion



Lab objectives

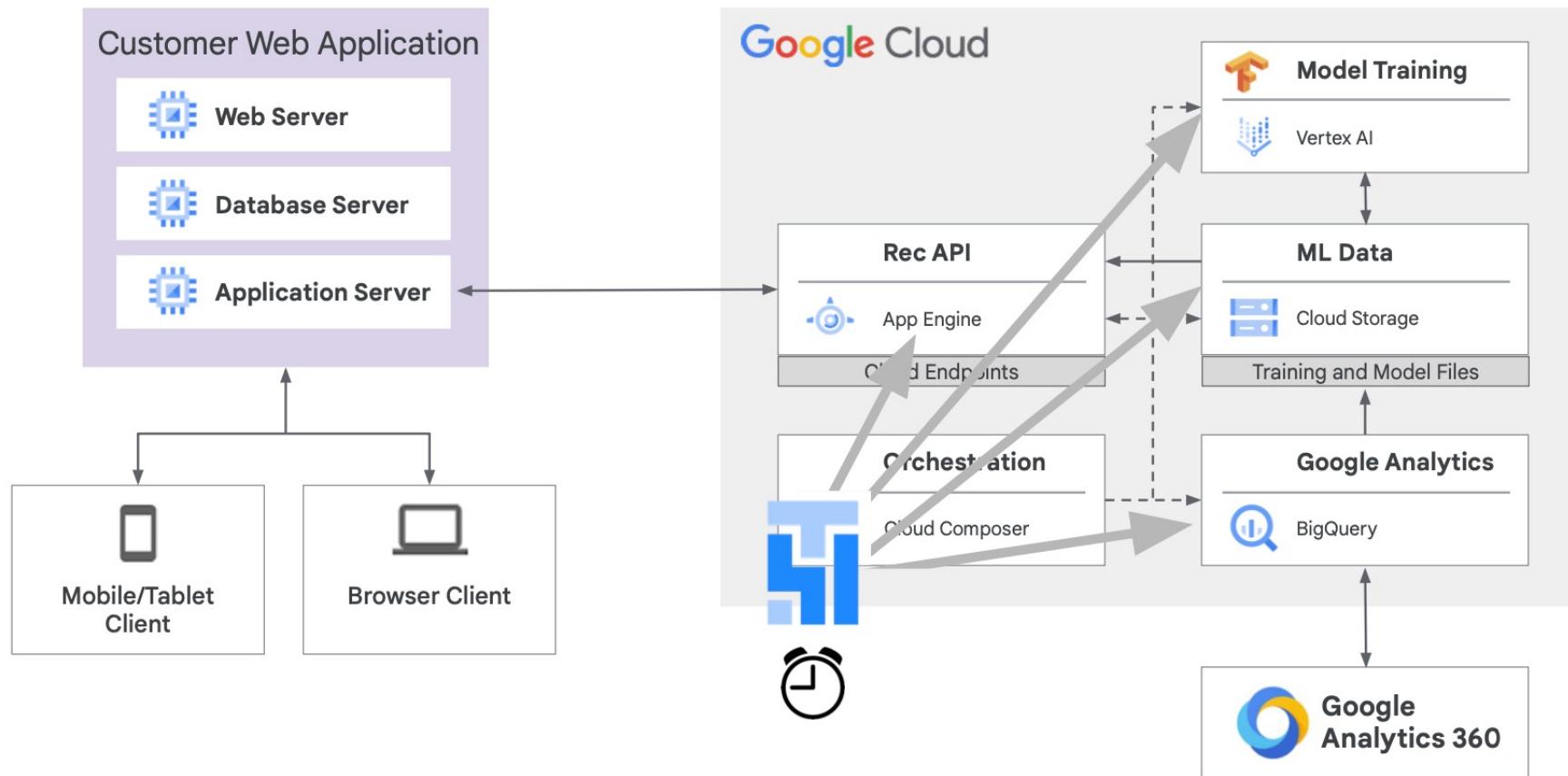
- 01 Connect Cloud Data Fusion to a couple of data sources
- 02 Apply basic transformations
- 03 Join two data sources
- 04 Write data to a sink



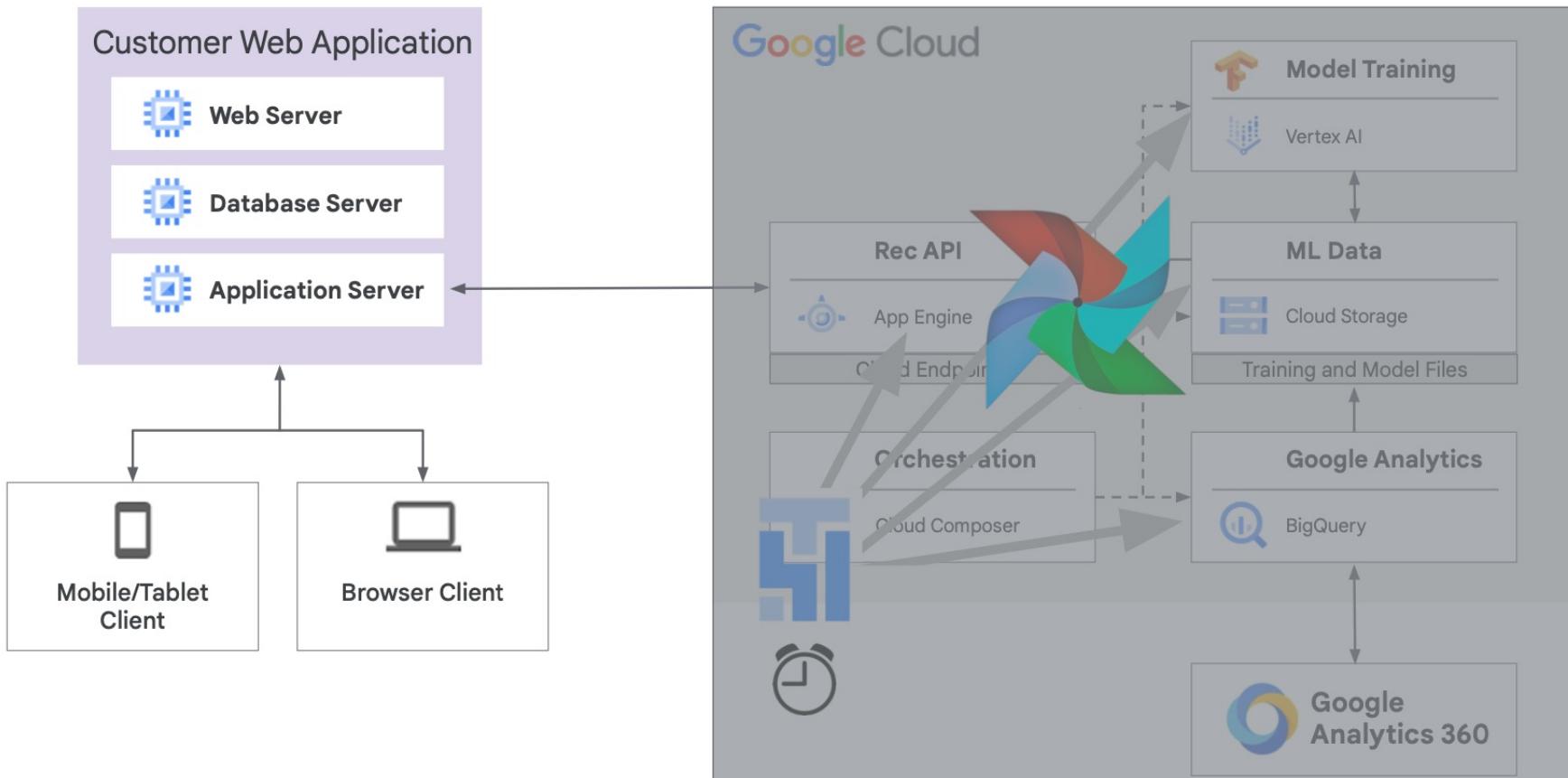


Orchestrating Work Between Google Cloud Services with Cloud Composer

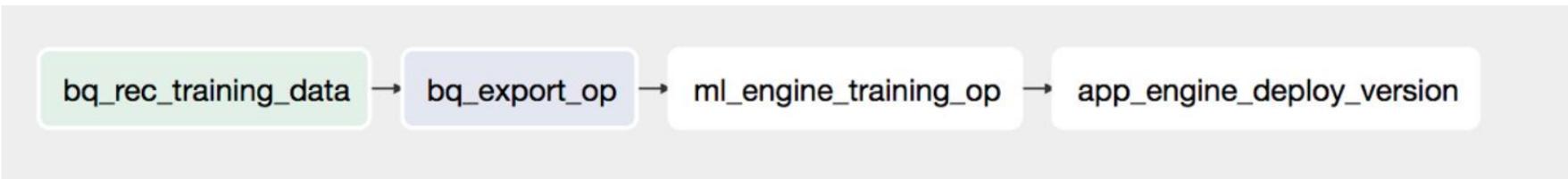
Cloud Composer orchestrates automatic workflows



Cloud Composer is managed Apache Airflow



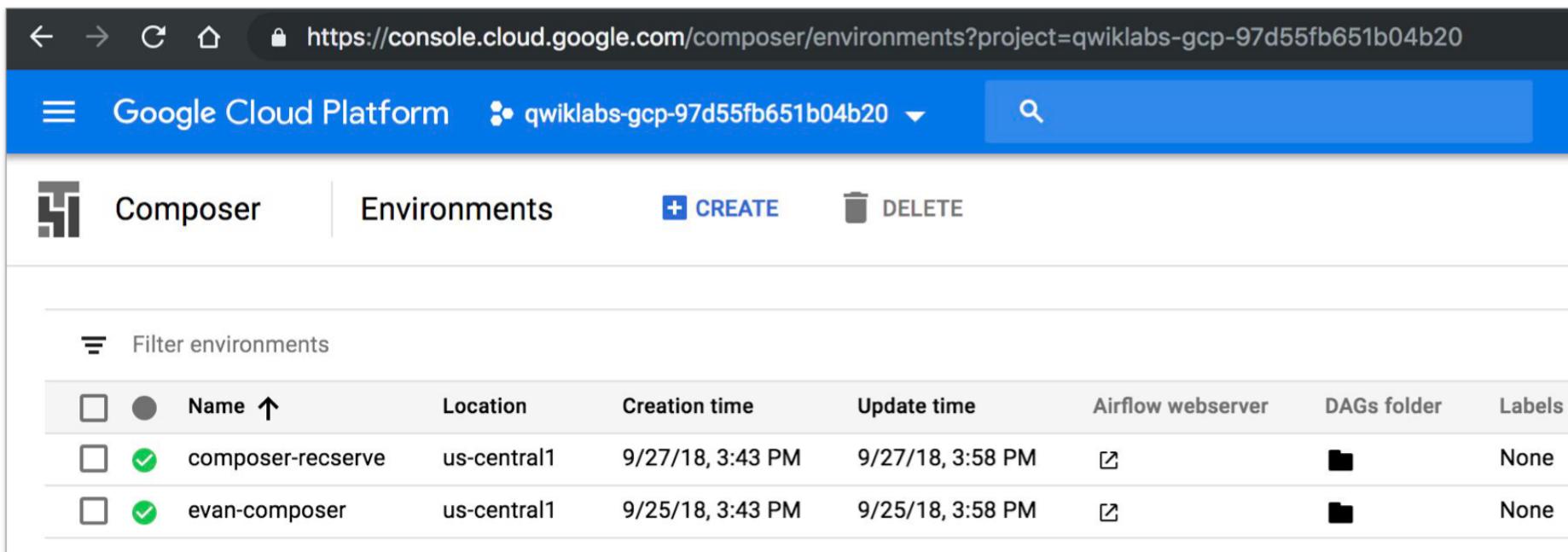
Use Apache Airflow DAGs to orchestrate Google Cloud services



DAG = Directed Acyclic Graph



Cloud Composer creates managed Apache Airflow environments



The screenshot shows the Google Cloud Platform interface for managing Apache Airflow environments. The URL in the browser bar is <https://console.cloud.google.com/composer/environments?project=qwiklabs-gcp-97d55fb651b04b20>. The page title is "Google Cloud Platform" and the project is "qwiklabs-gcp-97d55fb651b04b20". The main navigation tabs are "Composer" and "Environments", with "Environments" selected. Below the tabs are "CREATE" and "DELETE" buttons. A "Filter environments" section is present. The table lists two environments:

	Name ↑	Location	Creation time	Update time	Airflow webserver	DAGs folder	Labels
<input type="checkbox"/>	<input checked="" type="checkbox"/> composer-recserve	us-central1	9/27/18, 3:43 PM	9/27/18, 3:58 PM			None
<input type="checkbox"/>	<input checked="" type="checkbox"/> 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 Environments page. At the top, there's a navigation bar with a back arrow, forward arrow, refresh button, and a URL field containing <https://console.cloud.google.com/composer/environments?project=qwiklabs-gcp-97d55fb651b04b20>. Below the navigation is the Google Cloud Platform header with the project name `qwiklabs-gcp-97d55fb651b04b20`. The main content area has tabs for **Composer** and **Environments**, with **CREATE** and **DELETE** buttons. A filter section labeled "Filter environments" is present. The table lists two environments:

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

Two specific columns are highlighted with green boxes and circled with numbers: **1** points to the "Airflow webserver" column, and **2** points to the "DAGs folder" column.

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	Encryption
<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	Google-managed key
<input type="checkbox"/> simple.py	2.51 KB	text/x-python-script	Multi-Regional	10/1/18, 1:10 PM	Not public	Google-managed key

Airflow workflows are written in Python

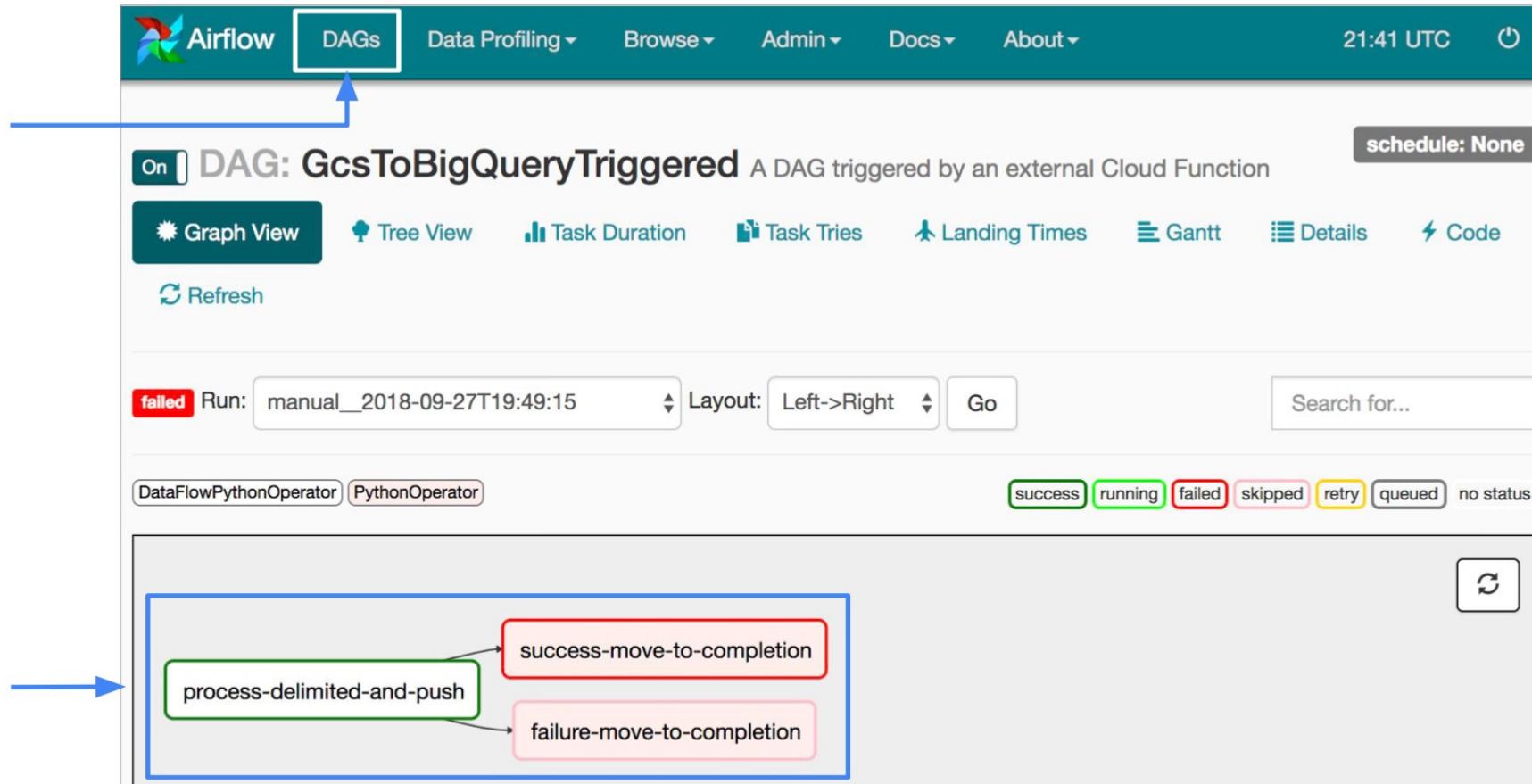
The diagram illustrates the relationship between a Google Cloud Storage bucket and an Airflow DAG Python file. On the left, a screenshot of the Google Cloud Storage interface shows a bucket named "us-central1-evan-composer-0e85530c-bucket" containing a folder "dags". Inside "dags" are two files: "simple_load_dag.py" (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 highlighted file 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

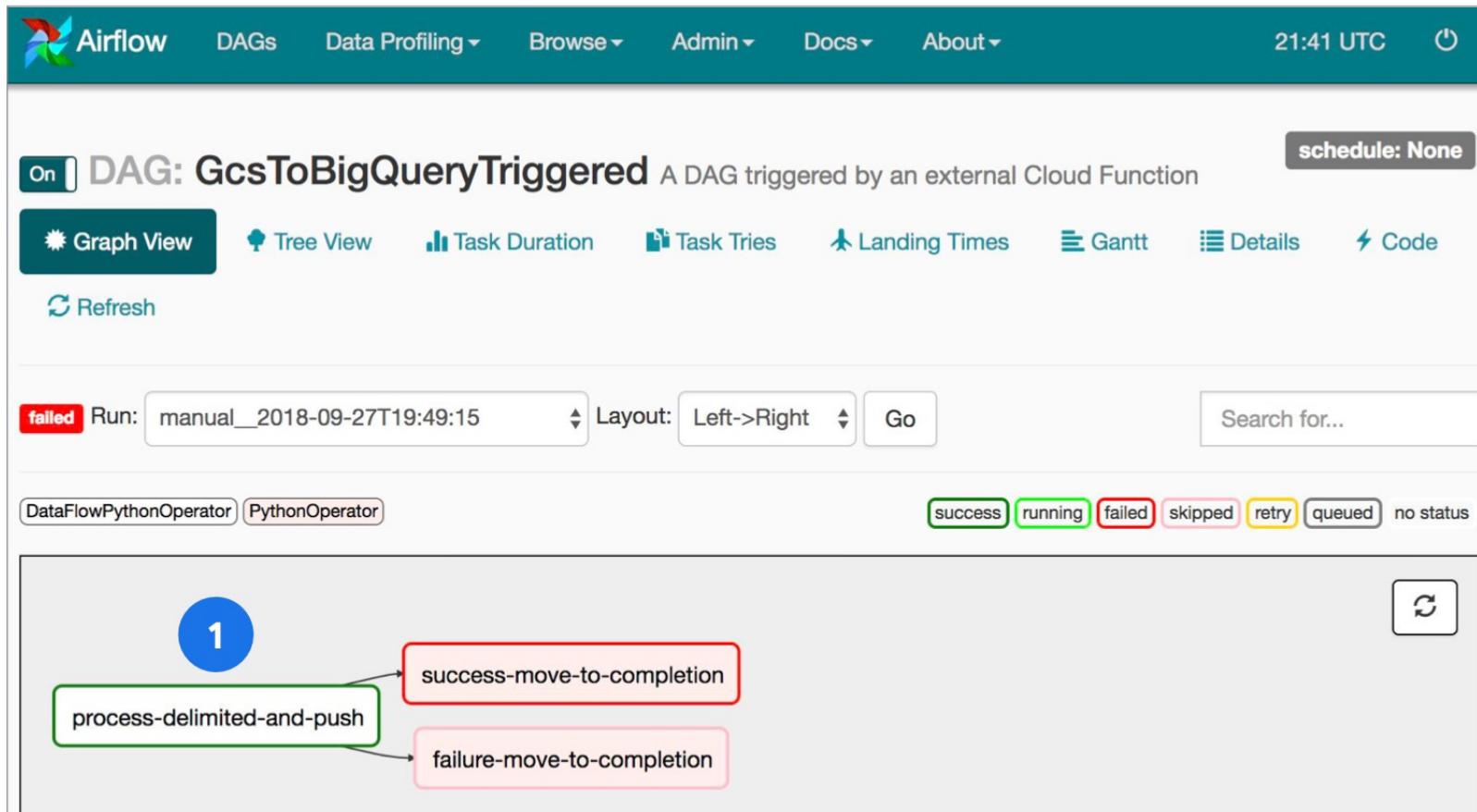
The screenshot shows the Airflow web server UI for the DAG: GcsToBigQueryTriggered. The top navigation bar includes links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, About, the current time (21:41 UTC), and a power icon.

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.

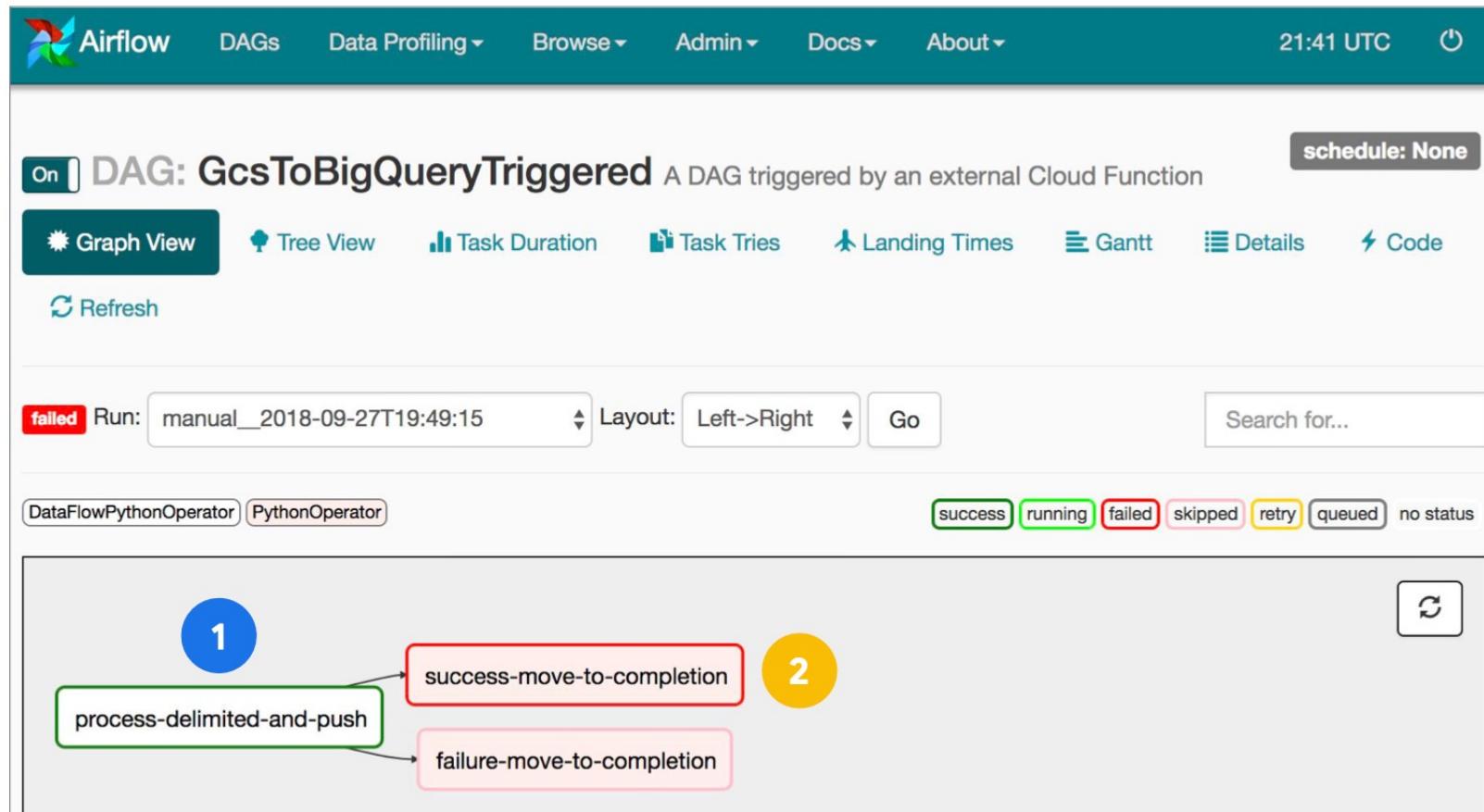
Below the navigation is a search bar with filters for Run (set to manual_2018-09-27T19:49:15) and Layout (set to Left->Right). There is also a "Go" button and a "Search for..." input field.

The central area shows the DAG structure. It includes two operator types: DataFlowPythonOperator and PythonOperator. Status indicators at the top right show success, running, failed, skipped, retry, queued, and no status. The DAG visualization shows a task named "process-delimited-and-push" with two outgoing arrows labeled "success-move-to-completion" and "failure-move-to-completion". A refresh icon is located in the top right corner of the DAG view.

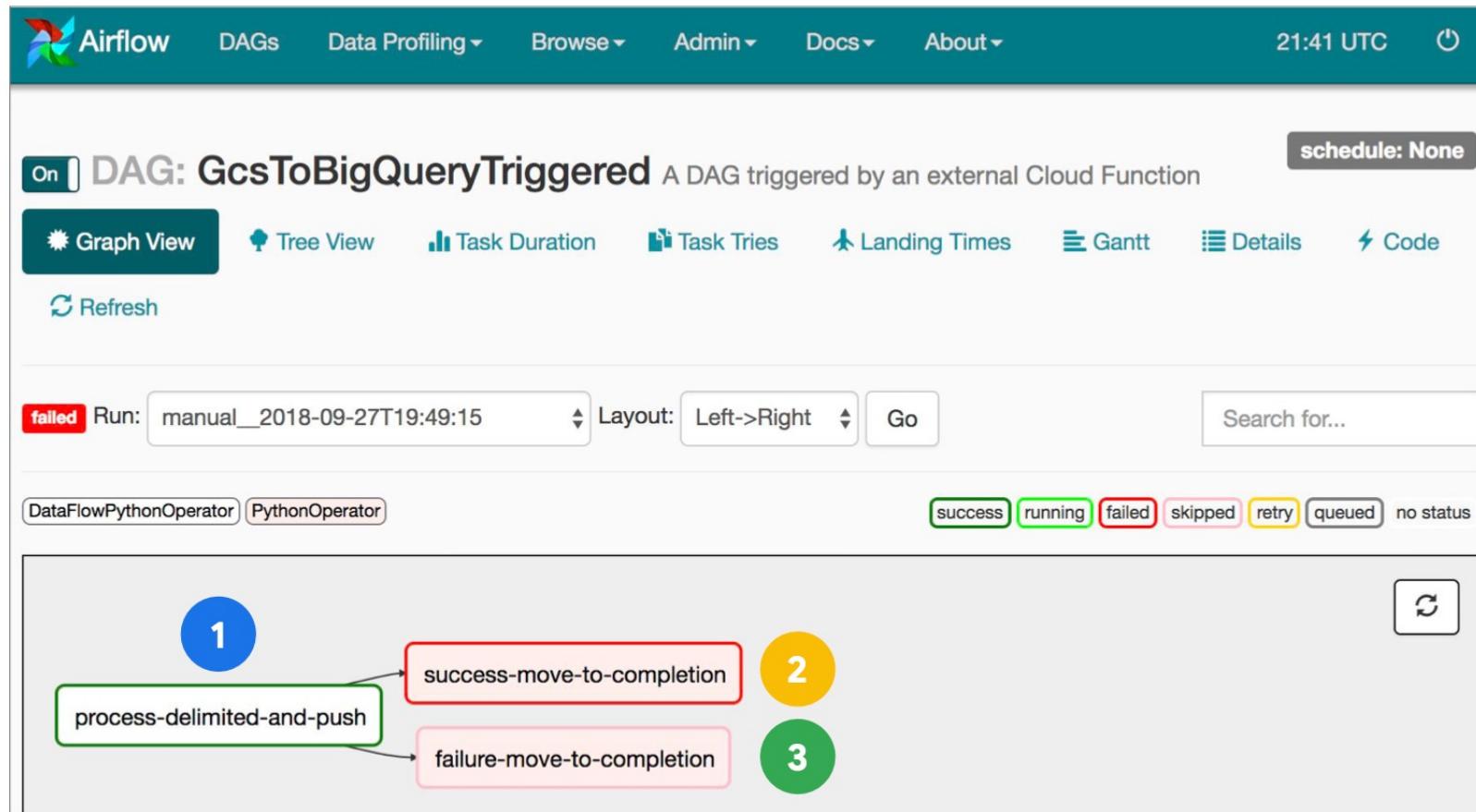
The Python file creates a DAG



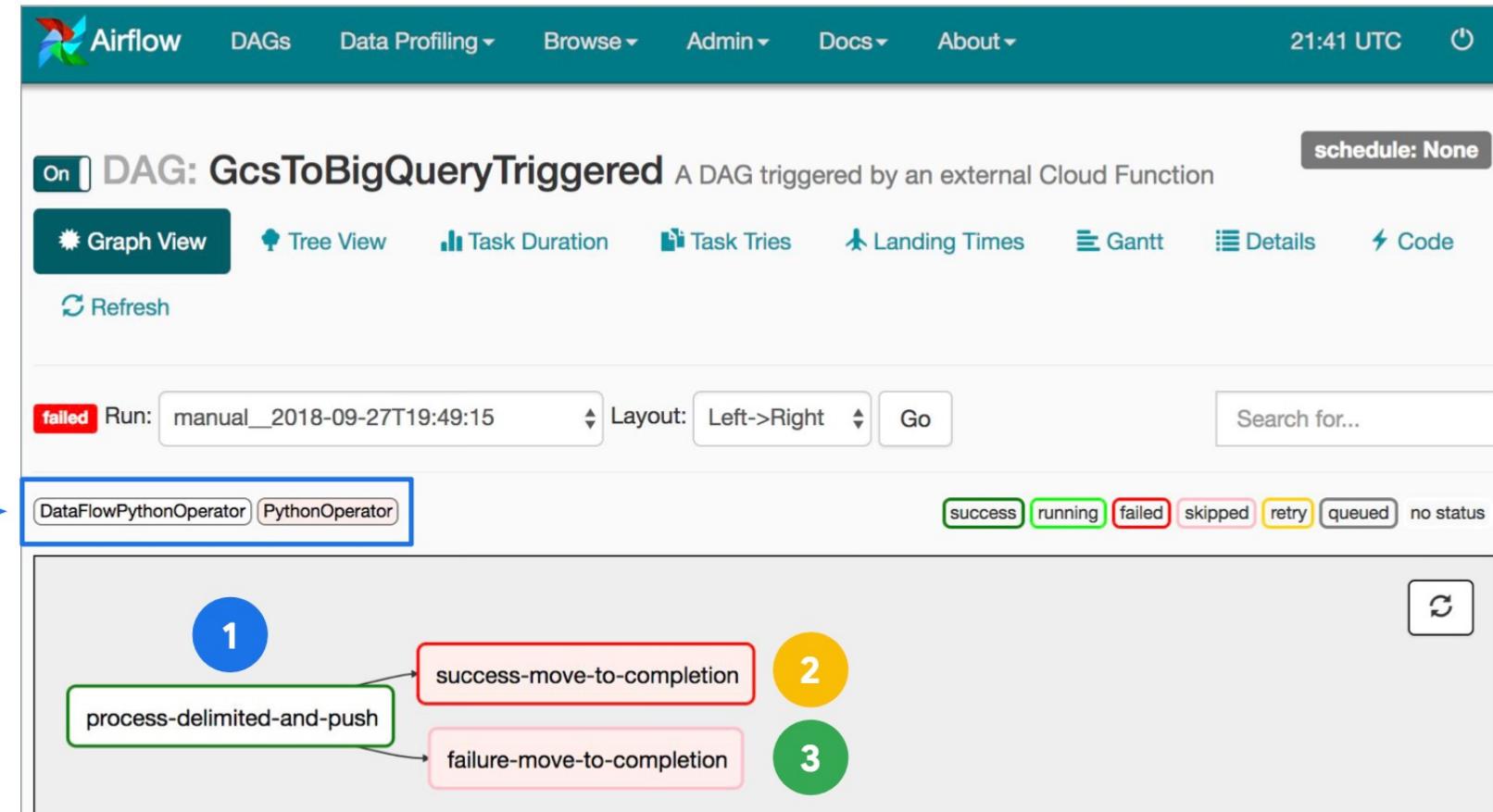
The Python file creates a DAG



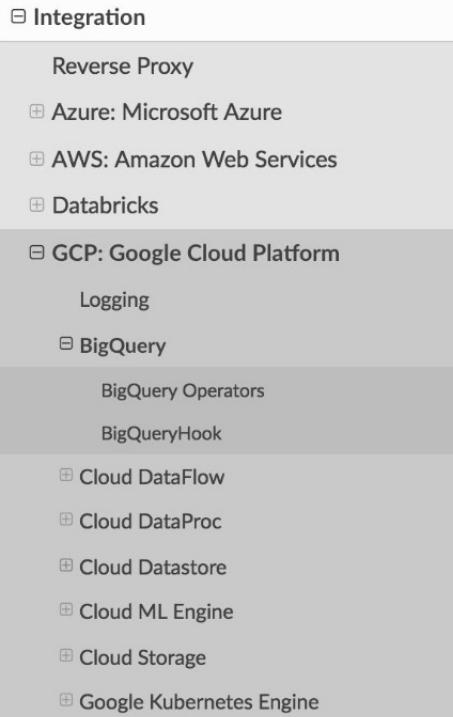
The Python file creates a DAG



The Python file creates a DAG



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

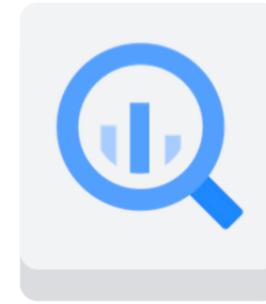
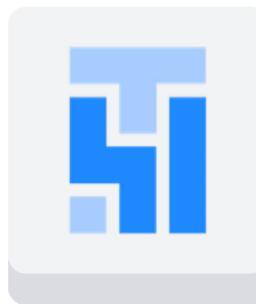
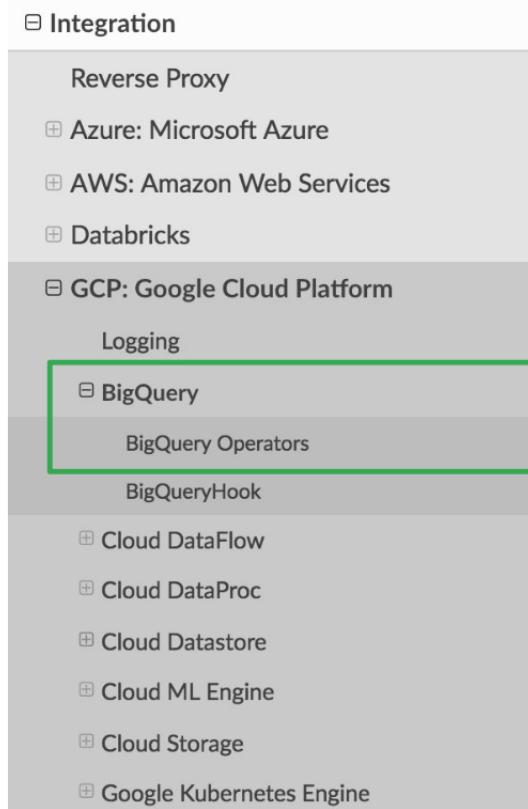


View all Google Cloud services that Airflow can orchestrate:

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

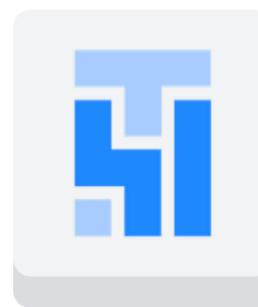
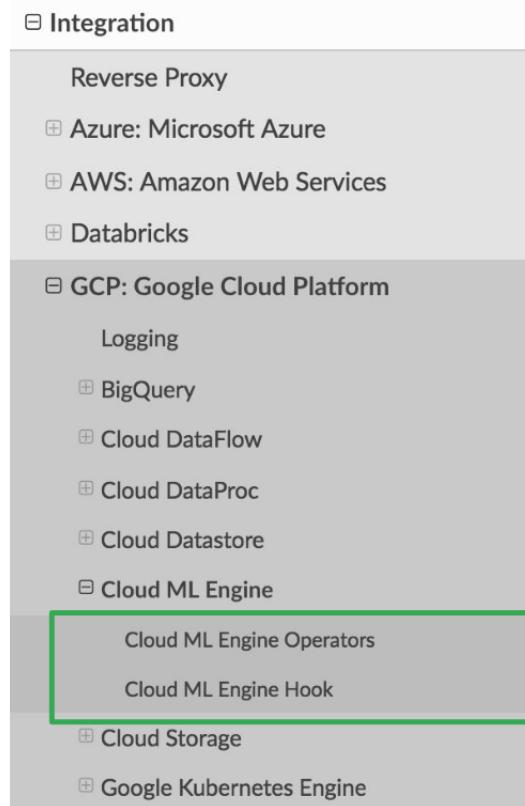
+ sample code

BigQuery Operators are popular for updating your ML training dataset



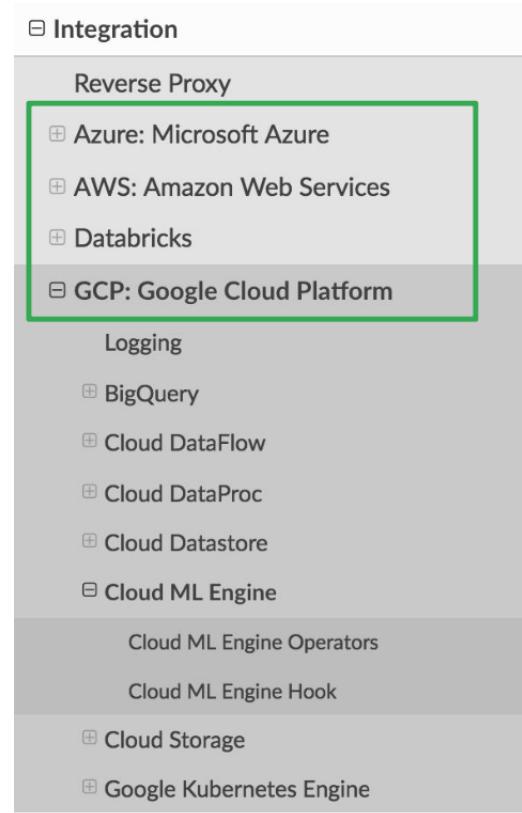
`BigQueryOperator`
`BigQueryCheckOperator`
`BigQueryIntervalCheckOperator`
`BigQueryCreateEmptyTableOperator`
`BigQueryCreateExternalTableOperator`
`BigQueryDeleteDatasetOperator`
`BigQueryToBigQueryOperator`
`BigQueryToCloudStorageOperator`

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

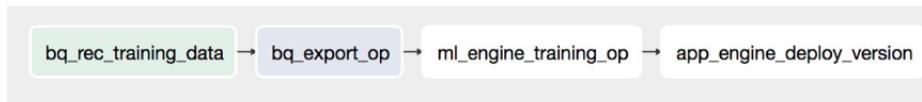


`MLEngineBatchPredictionOperator`
`MLEngineModelOperator`
`MLEngineTrainingOperator`
`MLEngineVersionOperator`

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

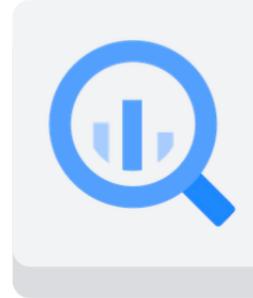
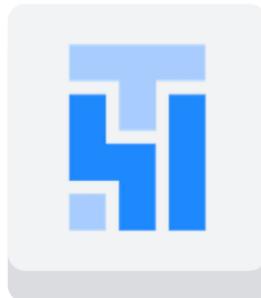


Example: Common Machine Learning workflow DAG Operators



```
# 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)
```

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',
    bql="""
        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',
    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)
```

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)
```

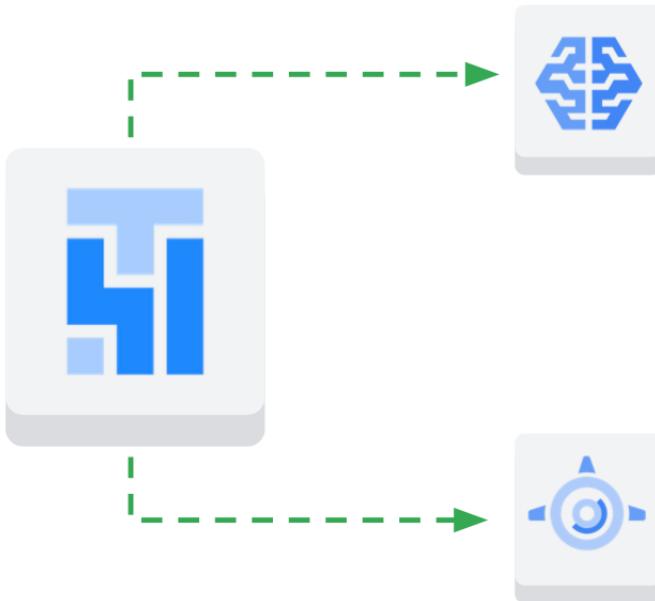
You could even set a rolling window with macros

```
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(
    )

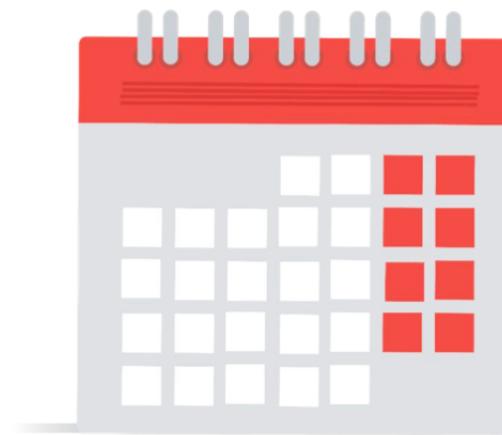
# BigQuery training data export to GCS
t2 = BigQueryToCloudStorageOperator(
    )

# 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

The screenshot shows the Airflow web interface with a teal header bar. The header includes the Airflow logo, navigation links for DAGs, Data Profiling, Browse, Admin, Docs, and About, and a timestamp of 20:14 UTC. A power icon is also present in the top right.

The main content area is titled "DAGs" and features a search bar labeled "Search:". Below the search bar is a table with the following columns: DAG, Schedule, Owner, Recent Tasks, Last Run, DAG Runs, and Links.

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

At the bottom of the table, a message indicates "Showing 1 to 2 of 2 entries".

Why is this DAG missing a schedule?

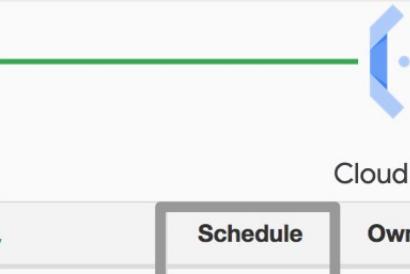
DAGs

Search:

	i	DAG	Schedule	Owner	Recent Tasks i	Last Run i	DAG Runs i	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	2018-02-21 00:00	40 12

Showing 1 to 2 of 2 entries

Option 1: Event-driven scheduling with Cloud Functions



DAGs

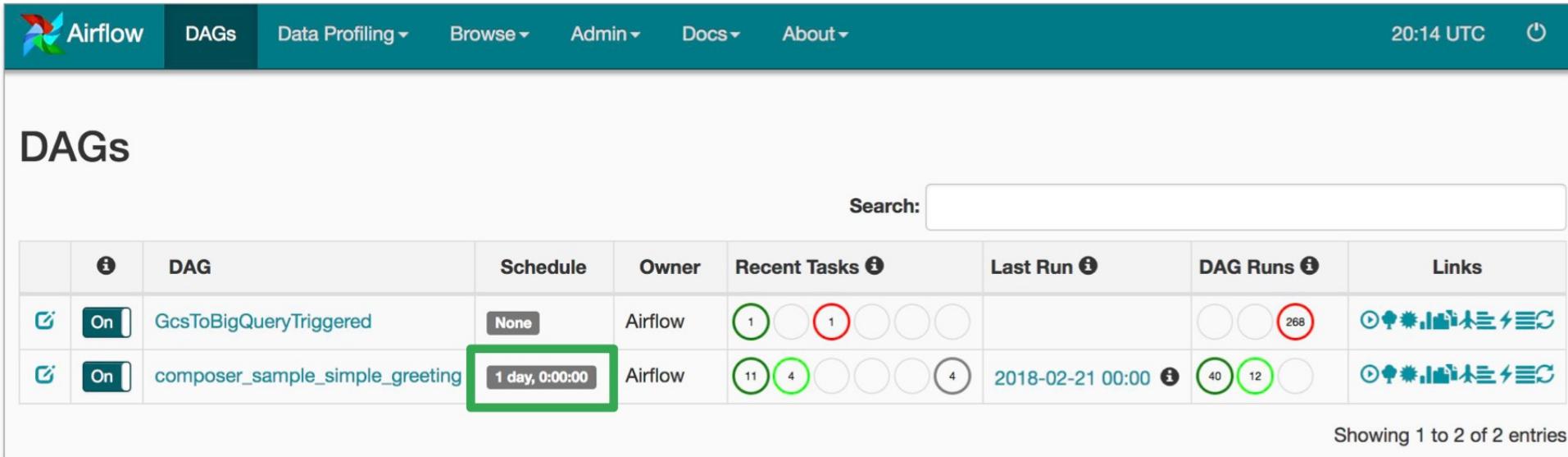
Cloud Functions

Search:

	i	DAG	Schedule	Owner	Recent Tasks i	Last Run i	DAG Runs i	Links
<input checked="" type="checkbox"/>	On	GcsToBigQueryTriggered	None	Airflow	1 1 1		268	1 2 3 4 5 6 7 8
<input checked="" type="checkbox"/>	On	composer_sample_simple_greeting	1 day, 0:00:00	Airflow	11 4	2018-02-21 00:00	40 12	1 2 3 4 5 6 7 8

Showing 1 to 2 of 2 entries

Option 2: Specify pipeline schedule_interval in your DAG



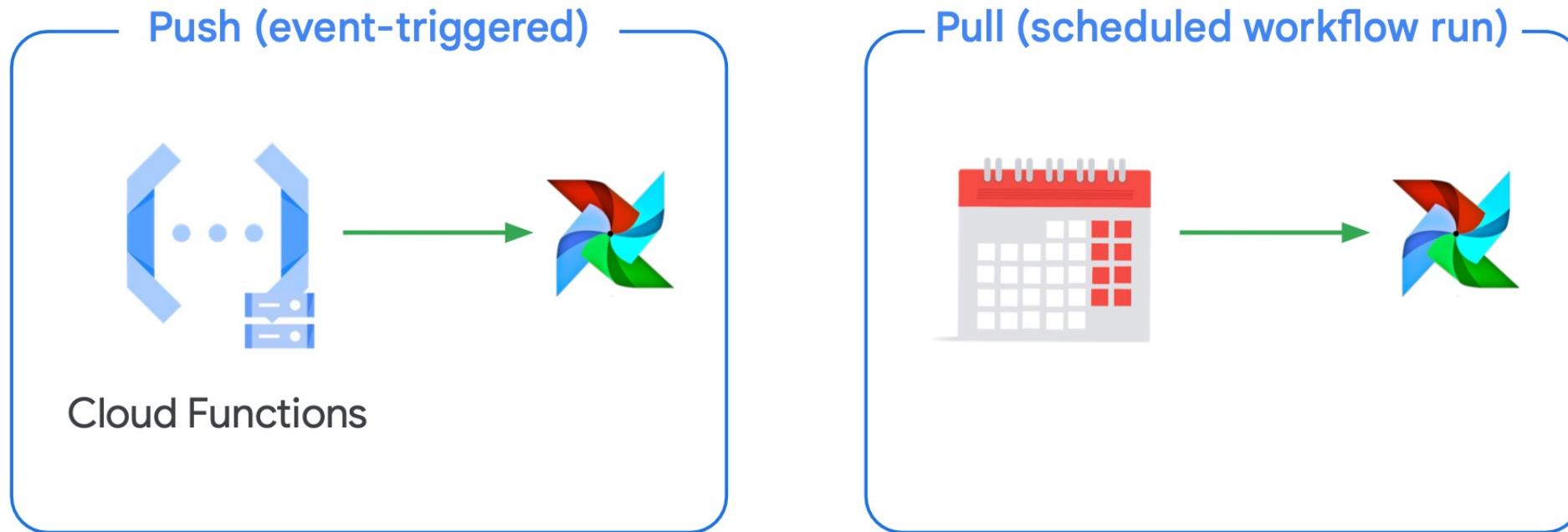
The screenshot shows the Airflow web interface with the 'DAGs' tab selected. The 'Search:' bar is empty. The table has the following columns: Action, DAG ID, DAG Name, Schedule, Owner, Recent Tasks, Last Run, DAG Runs, and Links. The 'composer_sample_simple_greeting' row is highlighted with a green box around its 'Schedule' cell, which contains '1 day, 0:00:00'. The 'GcsToBigQueryTriggered' row has 'None' in its 'Schedule' cell.

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

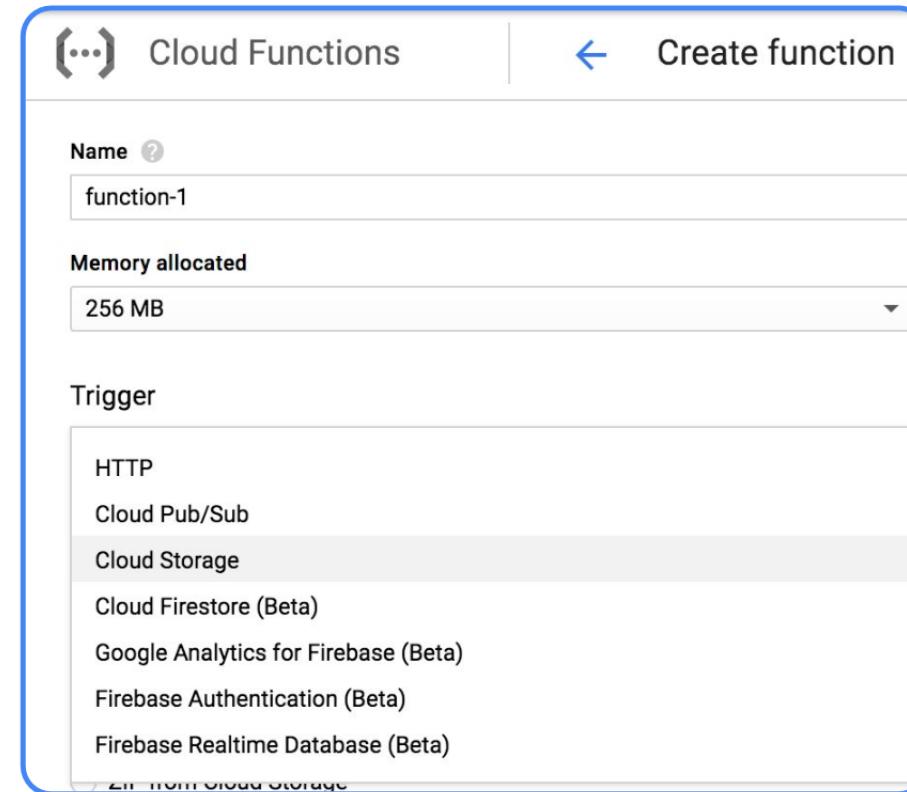
Showing 1 to 2 of 2 entries

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

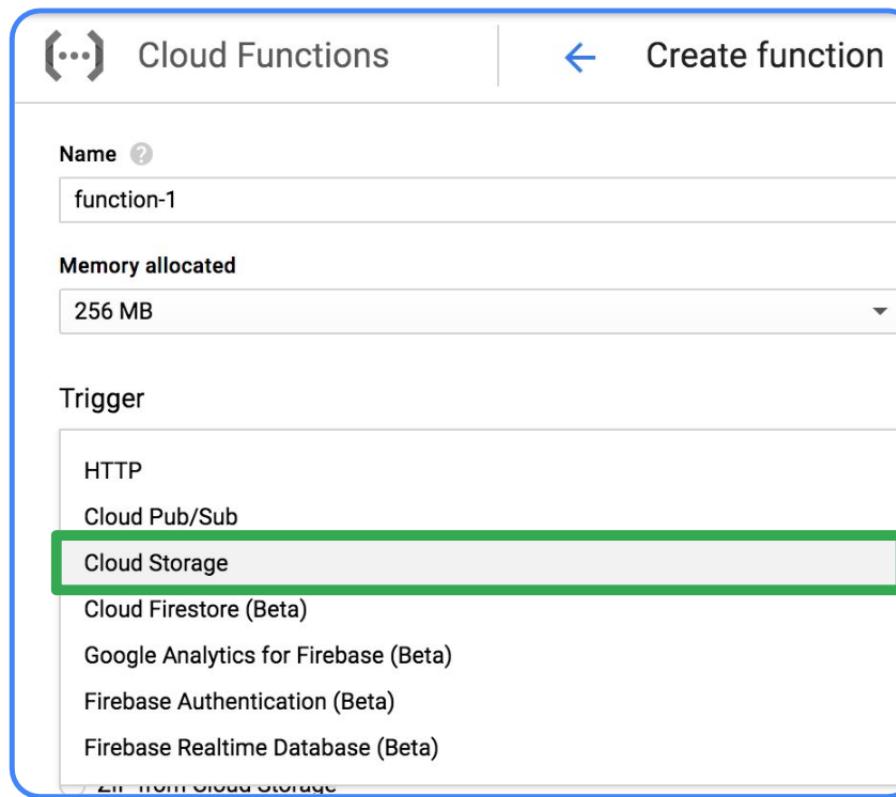
Two types of workflow ETL patterns



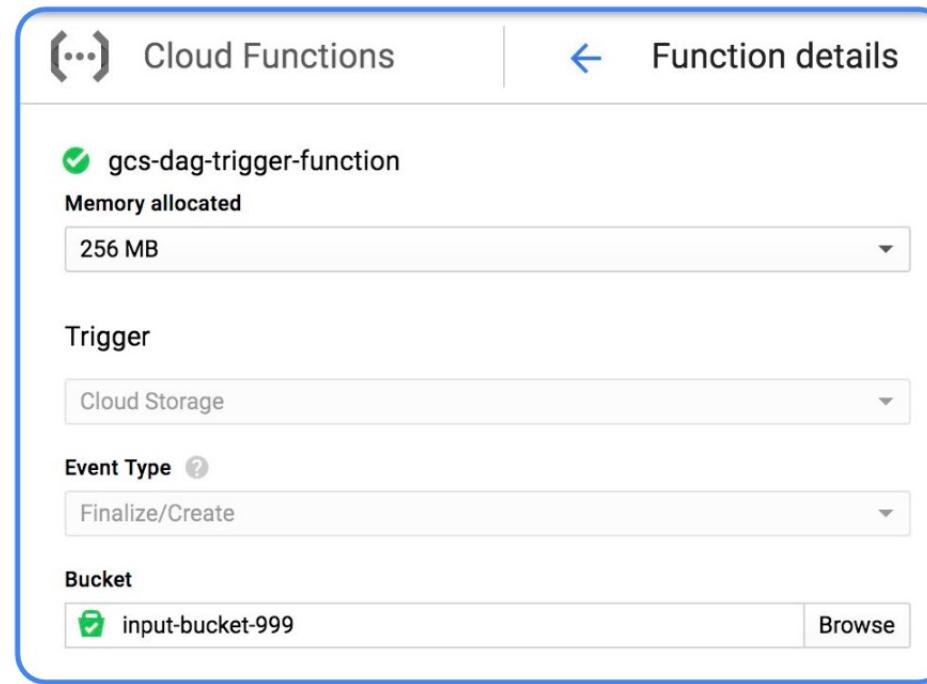
Use Cloud Functions to create an event-based push architecture



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

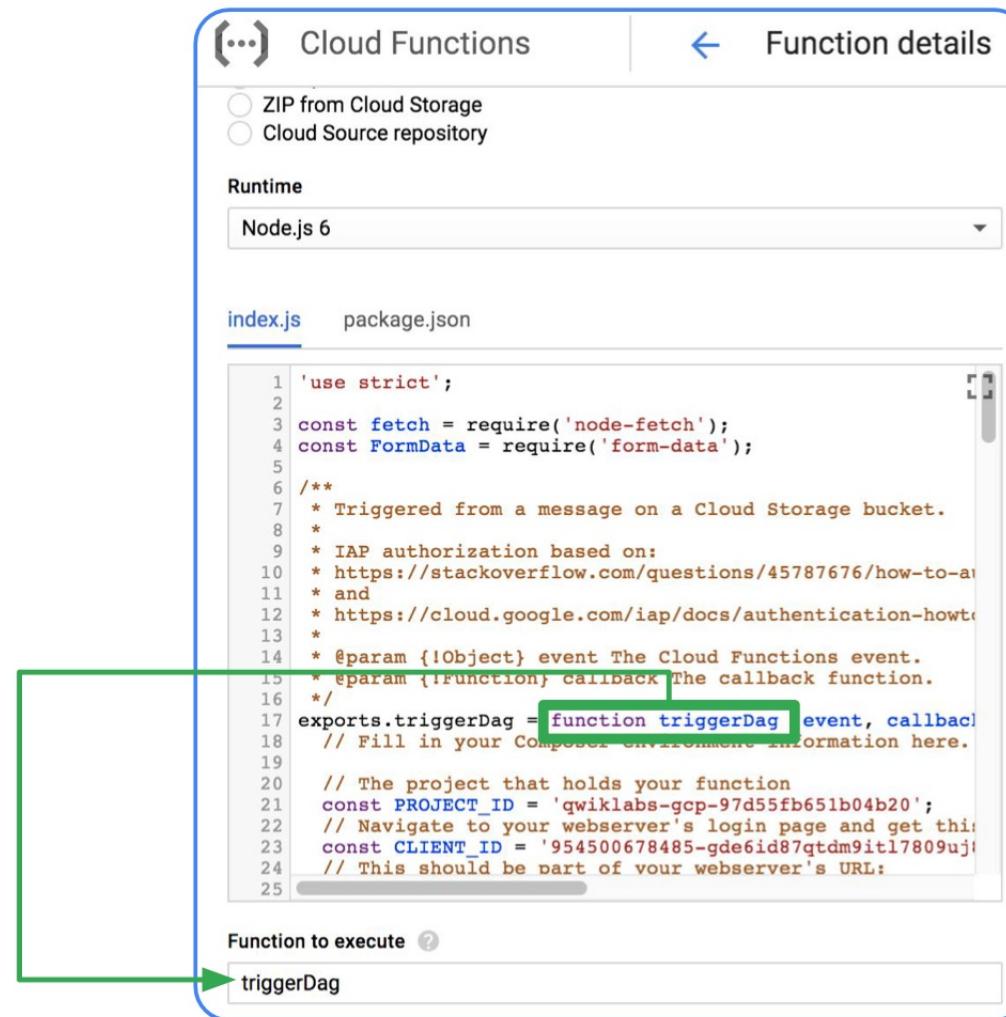
The diagram illustrates the execution flow of the Cloud Function code. It starts with the Cloud Function icon (1) pointing to the `index.js` file. The code defines a `triggerDag` function that triggers an Airflow DAG named `GcsToBigQueryTriggered`. The execution flow is as follows:

1. The Cloud Function icon points to the `index.js` file.
2. A green box highlights the URL construction: `const WEBSERVER_ID = 'b9...';` and `const DAG_NAME = 'GcsToBigQueryTriggered';`.
3. A green box highlights the DAG name: `const DAG_NAME = 'GcsToBigQueryTriggered';`.
4. A green box highlights the URL template: `const WEBSERVER_URL = `https://${WEBSERVER_ID}.appspot.com/api/experimental/dags/${DAG_NAME}/dag`;`.
5. A green box highlights the final code block: `authorizeIap(CLIENT_ID, PROJECT_ID, USER_AGENT)` followed by a promise chain.

```
index.js  package.json

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';
22   // Navigate to your webserver's login page and get this from the URL
23   const CLIENT_ID = '';
24   // This should be part of your webserver's URL:
25   // {tenant-project-id}.appspot.com
26   const WEBSERVER_ID = 'b9...';
27   // The name of the DAG you wish to trigger
28   const DAG_NAME = 'GcsToBigQueryTriggered';
29
30   // Other constants
31   const WEBSERVER_URL = `https://${WEBSERVER_ID}.appspot.com/api/experimental/dags/${DAG_NAME}/dag`;
32   const USER_AGENT = 'gcf-event-trigger';
33   const BODY = {'conf': JSON.stringify(event.data)};
34
35   // Make the request
36   authorizeIap(CLIENT_ID, PROJECT_ID, USER_AGENT)
37     .then(function iapAuthorizationCallback (iap) {
38       makeIapPostRequest(WEBSERVER_URL, BODY, iap.idToken, USER_AGENT, iap.jwt);
39     })
40     .then(_ => callback(null))
41     .catch(callback);
42   ;
43 }
```

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 code editor displays an index.js file 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-a
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 box highlights the `triggerDag` function name in the code editor. A green arrow points from this highlighted area to the 'Function to execute' input field at the bottom of the screen, 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 logo, the project name "qwiklabs-gcp-97d55fb651b04b20", and standard navigation icons. Below the header, there are tabs for "Cloud Functions" and "Overview", along with a "CREATE FUNCTION" button and other management icons. A search bar labeled "Filter functions" is present. The main table lists one function:

Name	Region	Trigger	Runtime	Memory allocated	Executed function
<input checked="" type="checkbox"/> 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
- Toolbar:** List (273), Create, Add Filter, With selected, Search, and a Reset Filters button.
- Filter Bar:** A dropdown for "Dag Id" set to "equals" with the value "composer_sample_simple_greet".
- Table:** A grid showing six historical dag runs for the "composer_sample_simple_greeting" DAG. Each row includes a checkbox, a edit icon, the state ("success"), the dag id, the execution date, the run id, and an external trigger icon.

		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

DAGs

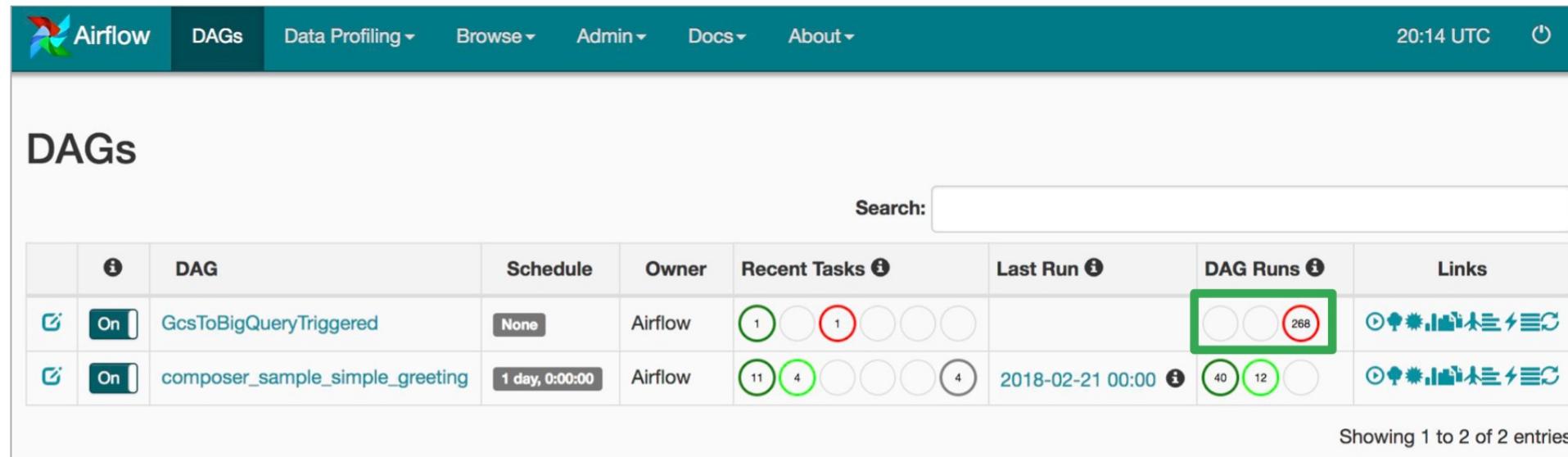
Search:

	i	DAG	Schedule	Owner	Recent Tasks i	Last Run i	DAG Runs i	Links
<input checked="" type="checkbox"/>	On	GcsToBigQueryTriggered	None	Airflow	1 1 1 1 1 1		1 1 268	1 2 3 4 5 6 7 8
<input checked="" type="checkbox"/>	On	composer_sample_simple_greeting	1 day, 0:00:00	Airflow	11 4 1 1 1 1 4	2018-02-21 00:00 i	40 12 1	1 2 3 4 5 6 7 8

Showing 1 to 2 of 2 entries

Which pipeline is healthier?

Quickly gauge pipeline health of DAG runs



DAGs

Search:

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

Showing 1 to 2 of 2 entries

Quickly gauge pipeline health of DAG runs

On DAG: **GcsToBigQueryTriggered** A DAG triggered by an external Cloud Function schedule: None

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh

Run: manual_2018-09-27T19:49:15 Layout: Left->Right Go Search for...

DataFlowPythonOperator PythonOperator success running failed skipped retry queued no status

process-delimited-and-push → success-move-to-completion
process-delimited-and-push → failure-move-to-completion

Monitor and troubleshoot Airflow step errors in logs

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

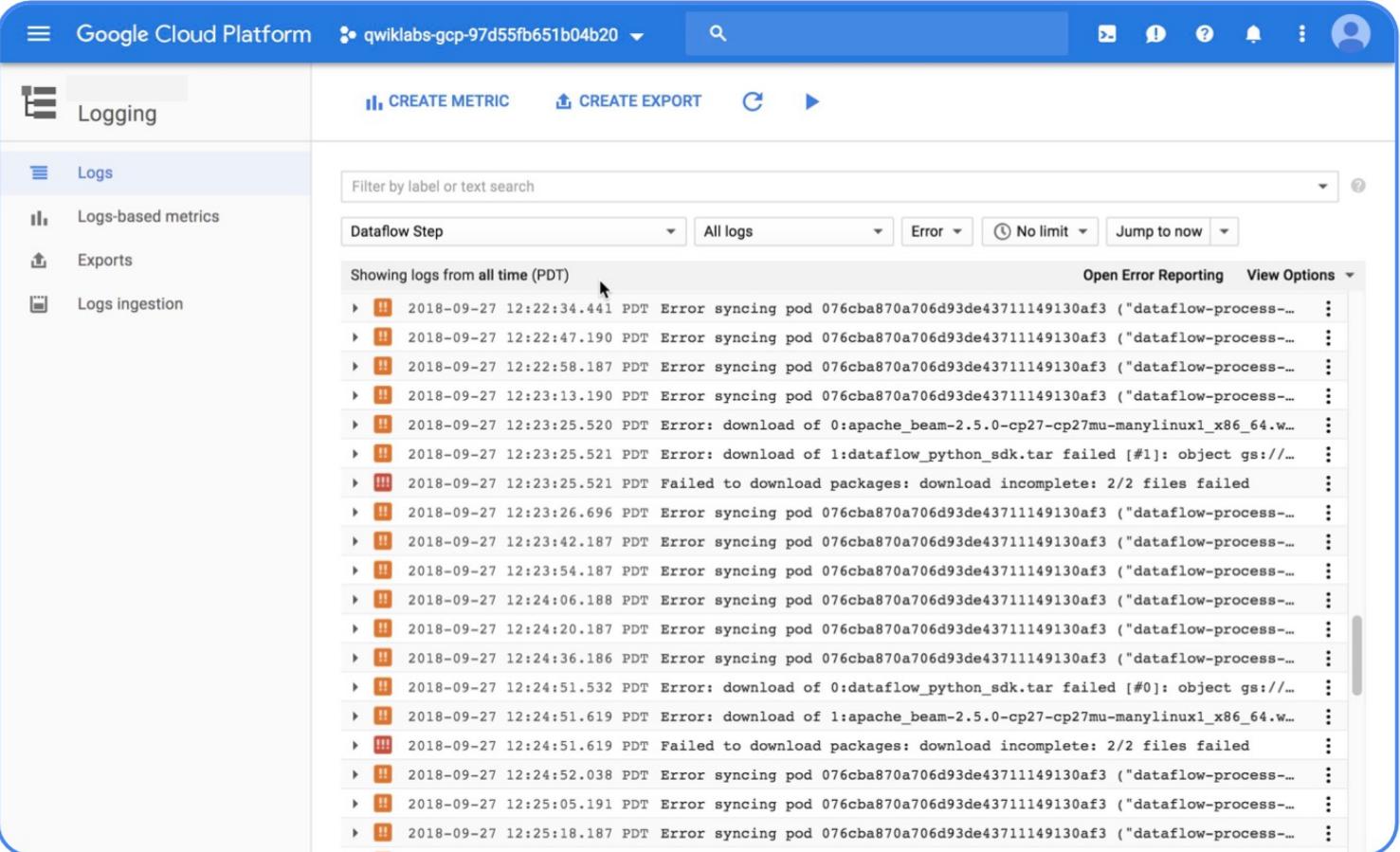
- DAG:** GcsToBigQueryTriggered (status: On)
- Schedule:** None
- Task Instance:** success-move-to-completion (2018-09-27 19:49:15)
- Log Tab:** Selected
- Log by attempts:** Attempt 1
- Log Content:** The log shows the task starting and attempting to execute a Python operator. It includes INFO messages from the airflow-worker and base_task_runner, and an error message indicating an invalid bucket name.

```
*** Reading remote log from gs://us-central1-evan-composer-0e85530c-bucket/logs/GcsToBigQueryTriggered/success-move-to-comple
[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 :
[2018-09-27 20:03:38,751] {base_task_runner.py:115} INFO - Running: ['bash', '-c', u'airflow run GcsToBigQueryTriggered succe
[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 ex
```

```
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-9
99%2F/o/success%2F2018-09-27%2Fusa_names.csv?alt=json returned "Invalid bucket name: 'gs://output-bucket-999/'>
```

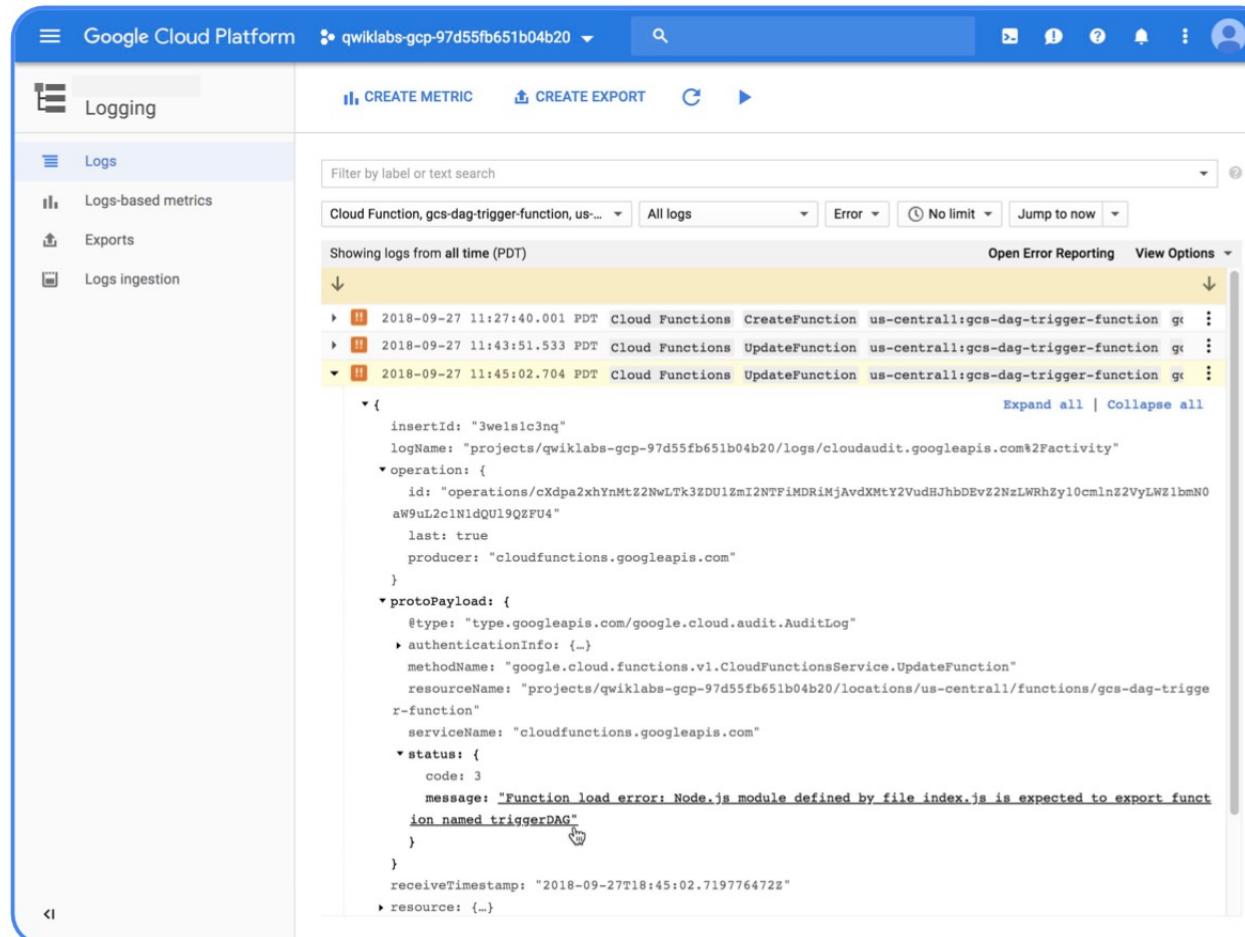
Monitor Dataflow health in Cloud Logging



The screenshot shows the Google Cloud Platform Logging interface. The left sidebar has 'Logs' selected. The main area displays a list of log entries under the heading 'Showing logs from all time (PDT)'. The logs are filtered by 'Dataflow Step', 'All logs', 'Error', and 'No limit'. The log entries show repeated errors related to syncing pods and package downloads.

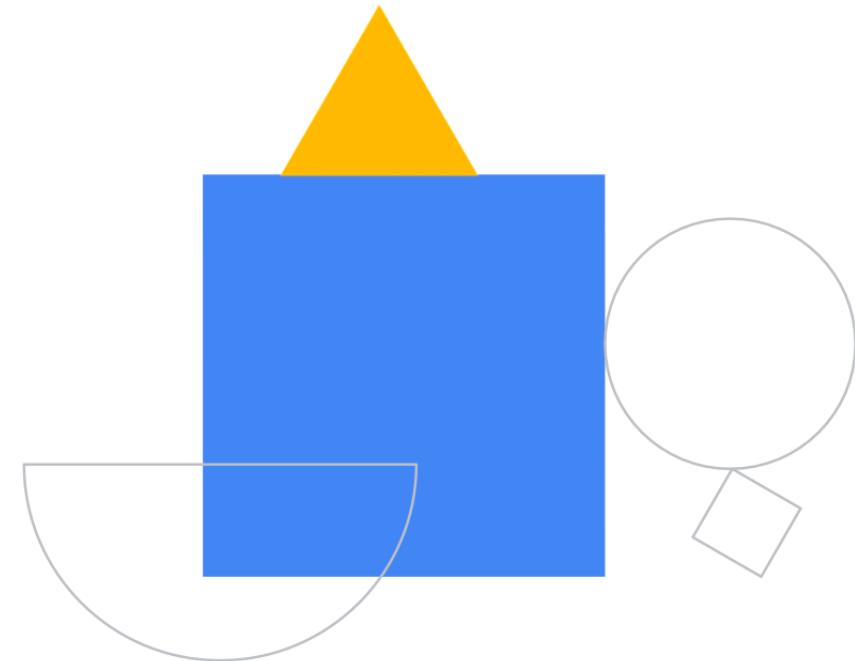
Date	Time	Message
2018-09-27	12:22:34.441	PDT Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:22:47.190	PDT Error syncing pod 076cba870a706d93de43711149130af3 ("dataflow-process-...")
2018-09-27	12:22:58.187	PDT Error syncing pod 076cba870a706d93de43711149130af3 ("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

