

📖 06c_customer_churn_airflow_execution.md

🔗 Customer Churn using Serverless Spark through Airflow.

Goal - Data Preparation and Model Training for Detecting Customer Churn Dataset.

Following are the lob modules:

- [1. Understanding Data](#)
- [2. Solution Diagram](#)
- [3. Uploading DAG files to DAGs folder](#)
- [4. Execution of Airflow DAG](#)
- [5. BQ output tables](#)
- [6. Logging](#)

🔗 1. Understanding Data

The dataset used for this project are [customer churn data](#) and [customer test data](#).

The dataset contains the following features:

- Churn - Binary field which represents customers who left/were retained within the last month
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they’ve been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

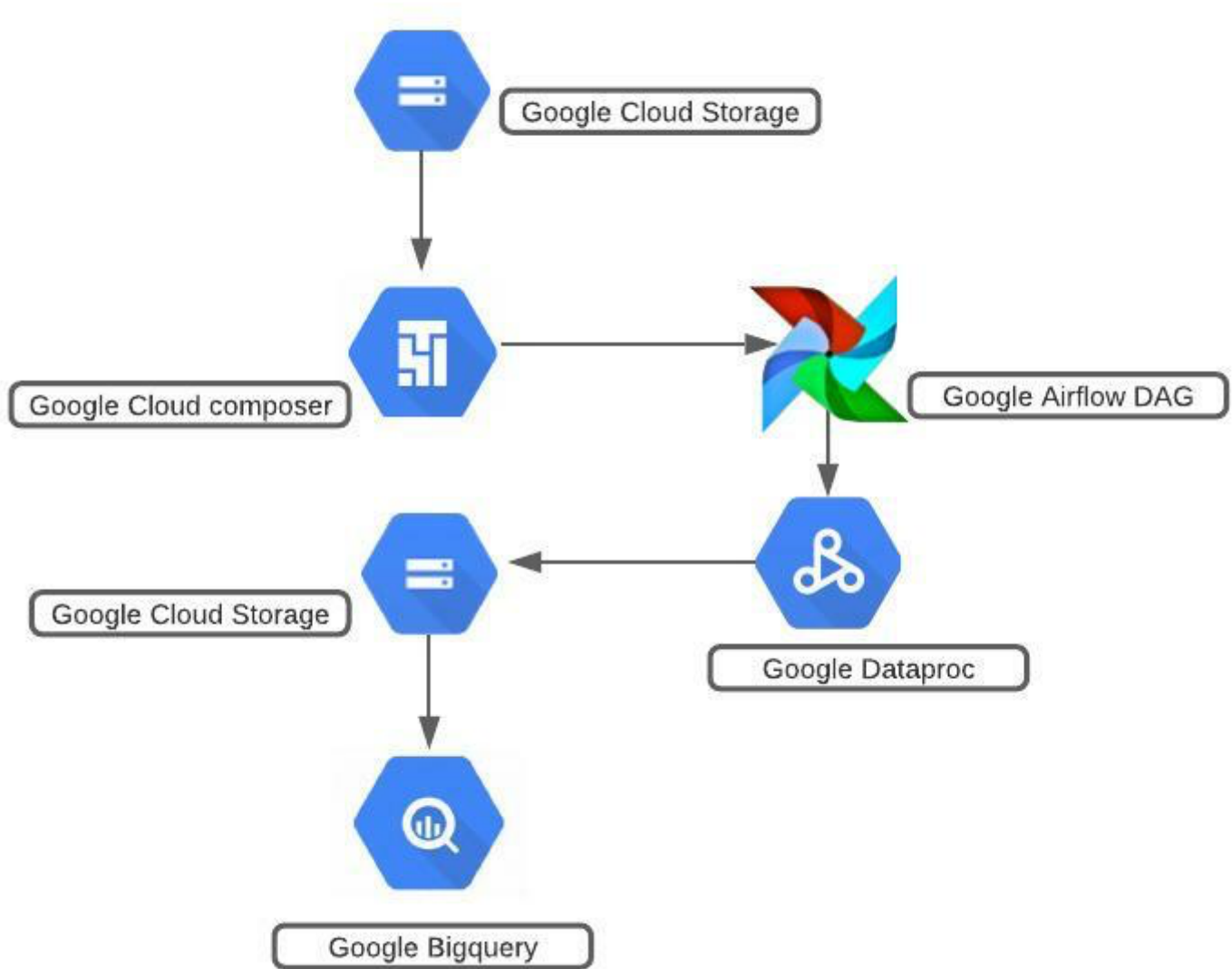
Note: The following features refer to these same-host connections.

- serror_rate
- rerror_rate
- same_srv_rate
- diff_srv_rate
- srv_count

Note: The following features refer to these same-service connections.

- srv_serror_rate
- srv_rerror_rate
- srv_diff_host_rate

2. Solution Diagram




Model Pipeline

The model pipeline involves the following steps:

- Create buckets in GCS
- Create Dataproc and Persistent History Server Cluster
- Copy the raw data files, PySpark and notebook files into GCS
- Create a Cloud Composer environment and Airflow jobs to run the serverless spark job
- Creating Google BigQuery tables with summary of anomalous cell towers

3. Uploading DAG files to DAGs folder

- From the code repository, download the file located at: **customer_churn>00-scripts>customer_churn_airflow.py**
- Rename file to <your_name_here>-customer_churn_airflow.py
- Open the file and replace your name on row 21
- Navigate to **Composer><composer_environment>**
- Next, navigate to **Environment Configuration>DAGs folder URI**
- Next, upload the DAG file to the GCS bucket corresponding to the **DAGs folder URI**

MONITORING	LOGS	DAGS	PREVIEW	ENVIRONMENT CONFIGURATION
<hr/>				
Name	serverless-spark-composer			
Location	us-central1			
Service account	198454710197-compute@developer.gserviceaccount.com			
Image version	composer-2.0.5-airflow-2.2.3 UPGRADE  New version available. Learn more			
Python version	3			
DAGs folder	gs://us-central1-serverless-spar-24f8ee61-bucket/dags			

Buckets > us-central1-serverless-spar-24f8ee61-bucket > dags

Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Subject to object ACLs	None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE

Buckets > us-central1-serverless-spar-24f8ee61-bucket > dags

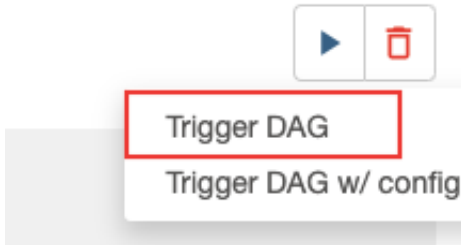
UPLOAD FILES
UPLOAD FOLDER
CREATE FOLDER
MANAGE HOLDS
DOWNLOAD
DELETE

4. Execution of Airflow DAG

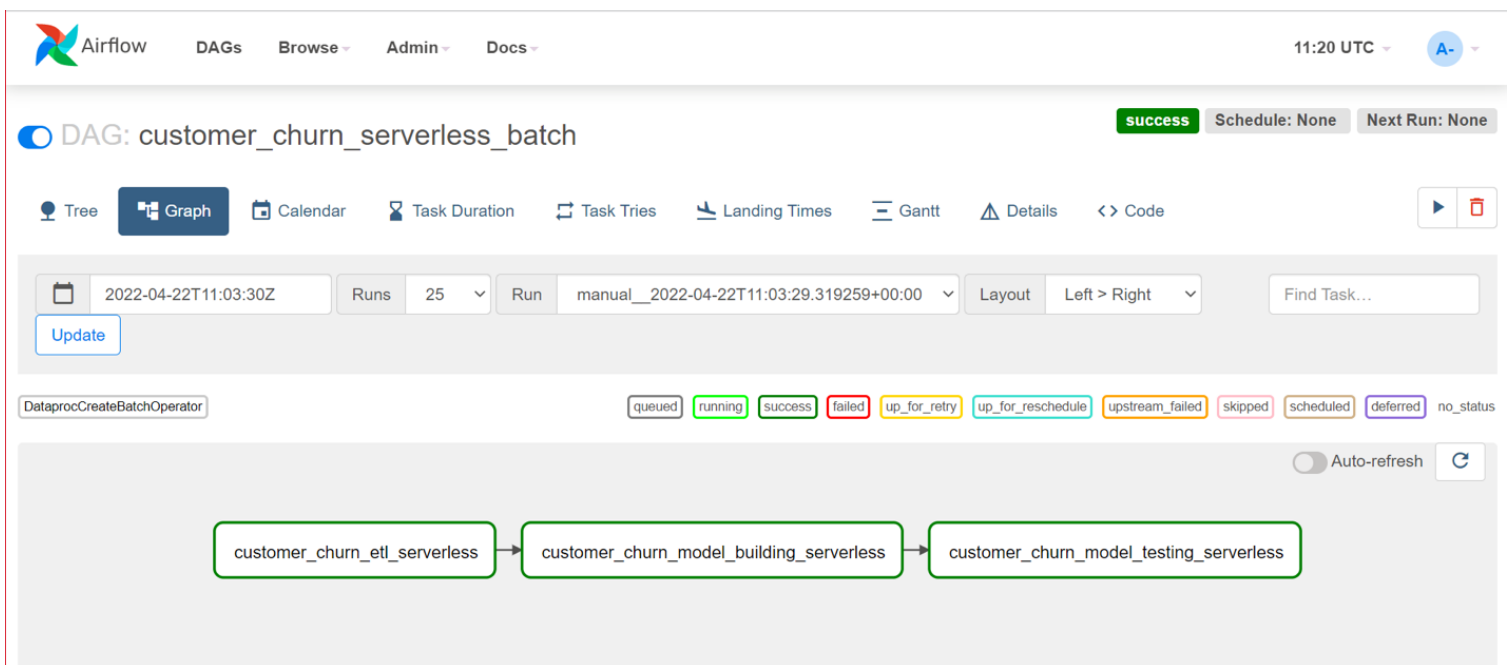
- Navigate to **Composer><your_environment>>Open Airflow UI**



- Once the Airflow UI opens, navigate to **DAGs** and open your respective DAG
- Next, trigger your DAG by clicking on the **Trigger DAG** button



- Once the DAG is triggered, the DAG can be monitored directly through the Airflow UI as well as the Dataproc>Serverless>Batches window



5. BQ output tables

Navigate to BigQuery Console, and check the **customer-churn** dataset.

Once the Airflow DAG execution is completed, four new tables '<your_name_here>_training_data', '<your_name_here>_test_data', '<your_name_here>_predictions_data' and '<your_name_here>_test_output' are created.

To view the data in these tables -

- Select the table from BigQuery Explorer by navigating 'project_id' > 'dataset' > 'table_name'
- Click on the **Preview** button to see the data in the table



Note: If the **Preview** button is not visible, run the below queries to view the data. However, these queries will be charged for the full table scan.

```
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_training_data` LIMIT 1000;
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_test_data` LIMIT 1000;
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_predictions_data` LIMIT 1000;
SELECT * FROM `<project_name>.<dataset_name>.<your_name_here>_test_output` LIMIT 1000;
```

The first screenshot shows a query being executed in the Grip interface. The query is: `SELECT * FROM `customer_churn`.`customer_churn`.`_training_data` LIMIT 1000`. The results are displayed in a table with 12 columns: Row, customerID, gender, SeniorCitizen, Partner, Dependents, tenure, tenure_group, PhoneService, MultipleLines, InternetService, and OnlineSecurity. The results show two rows of data.

Row	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	tenure_group	PhoneService	MultipleLines	InternetService	OnlineSecurity
1	0661-KQHNK	Female	0	true	true	6	Tenure_0-12	true	No	No	No
2	1269-FOYWN	Male	0	true	true	44	Tenure_24-48	true	No	No	No

The second screenshot shows a query being executed in the Grip interface. The query is: `SELECT * FROM `customer_churn1`.`customer_churn1`.`_test_data` LIMIT 1000`. The results are displayed in a table with 13 columns: Row, customerID, gender, SeniorCitizen, Partner, Dependents, tenure, tenure_group, PhoneService, MultipleLines, InternetService, OnlineSecurity, and OnlineBacku. The results show two rows of data.

Row	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	tenure_group	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBacku
1	0761-AETCS	Female	0	No	No	1	Tenure_0-12	Yes	No	No	No	No
2	1015-OWJKI	Male	0	No	No	1	Tenure_0-12	Yes	No	No	No	No

Note: Edit all occurrences of `<project_name>` and `<dataset_name>` to match the values of the variables `PROJECT_ID`, and `BQ_DATASET_NAME` respectively

6. Logging

6.1 Airflow logging

- To view the logs of any step of the DAG execution, click on the **>Log** button

[Instance Details](#) [Rendered](#) [Log](#) [All Instances](#) [Filter Upstream](#)

Download Log (by attempts):

1

Task Actions

[Ignore All Deps](#) [Ignore Task State](#) [Ignore Task Deps](#) [Run](#)

[Past](#) [Future](#) [Upstream](#) [Downstream](#) [Recursive](#) [Failed](#) [Clear](#)

[Past](#) [Future](#) [Upstream](#) [Downstream](#) [Mark Failed](#)

[Past](#) [Future](#) [Upstream](#) [Downstream](#) [Mark Success](#)

[Close](#)

6.2 Serverless Batch logs

Logs associated with the application can be found in the logging console under **Dataproc > Serverless > Batches > <batch_name>**.

You can also click on “View Logs” button on the Dataproc batches monitoring page to get to the logging page for the specific Spark job.

Logs Explorer

OPTIONS

REFINE SCOPE

Project

SHARE LINK

1:31:19 AM - 1:35:02 AM

LE

Query

Recent (5)

Saved (0)

Suggested (1)

Clear query

Save

Stream logs

Run query

resource.type="cloud_dataproc_batch" resource.labels.project_id="redacted" resource.labels.location="us-cen...

Edit query

Log fields

Histogram

Create metric

Create alert

Jump to now

More actions

Log fields

Search fields and values

RESOURCE TYPE

Cloud Dataproc Batch

Clear

SEVERITY

Default

severity

5,833

Info

235

Warning

6

LOG NAME

Query results

6,074 log entries

Download

SEVERITY	TIMESTAMP	PST	SUMMARY
> *	2022-03-02 01:34:39.579 PST		"WARNING: Use --illegal-access=warn to enable warnings of further illegal ...
> *	2022-03-02 01:34:39.580 PST		"WARNING: All illegal access operations will be denied in a future release"
> i	2022-03-02 01:34:40.000 PST		"Lease tasks stream ended with error (will restart): Status(code=ALREADY_E...
> !	2022-03-02 01:34:42.000 PST		"Input column Churn does not exist during transformation. Skip StringIndex...
> !	2022-03-02 01:34:43.000 PST		"Truncated the string representation of a plan since it was too large. Thi...
> i	2022-03-02 01:34:43.036 PST		{"@type": "type.googleapis.com/google.cloud.dataproc.logging.AutoscalerLog"...
> i	2022-03-02 01:34:43.449 PST		{"@type": "type.googleapis.com/google.cloud.dataproc.logging.AutoscalerLog"...
> i	2022-03-02 01:34:45.000 PST		"Lease tasks stream ended with error (will restart): Status(code=ALREADY_E...
> i	2022-03-02 01:34:45.000 PST		"Lease tasks stream ended with error (will restart): Status(code=ALREADY_E...

6.3 Persistent History Server logs

To view the Persistent History server logs, click the 'View History Server' button on the Dataproc batches monitoring page and the logs will be shown as below:

[←](#) batch-8096

[CLONE](#)

[DELETE](#)

[VIEW LOGS](#)

[REFRESH](#)

[VIEW SPARK HISTORY SERVER](#)

Batch ID

batch-8096

Batch UUID

1258322f-3f86-49cf-adc9-eaebb1c8df22

Resource type

Batch

Status

✓ Succeeded

MONITORING

DETAILS

 **History Server**

Event log directory: gs://bo_bucket_phs/phs/*/spark-job-history

Last updated: 2022-03-16 14:24:27

Client local time zone: America/Los_Angeles

Search:

Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.2.1	app-20220316212124-0000		10.0.0.8	2022-03-16 14:21:21	2022-03-16 14:22:01	40 s	spark	2022-03-16 14:22:01	Download

Showing 1 to 1 of 1 entries

[Show incomplete applications](#)