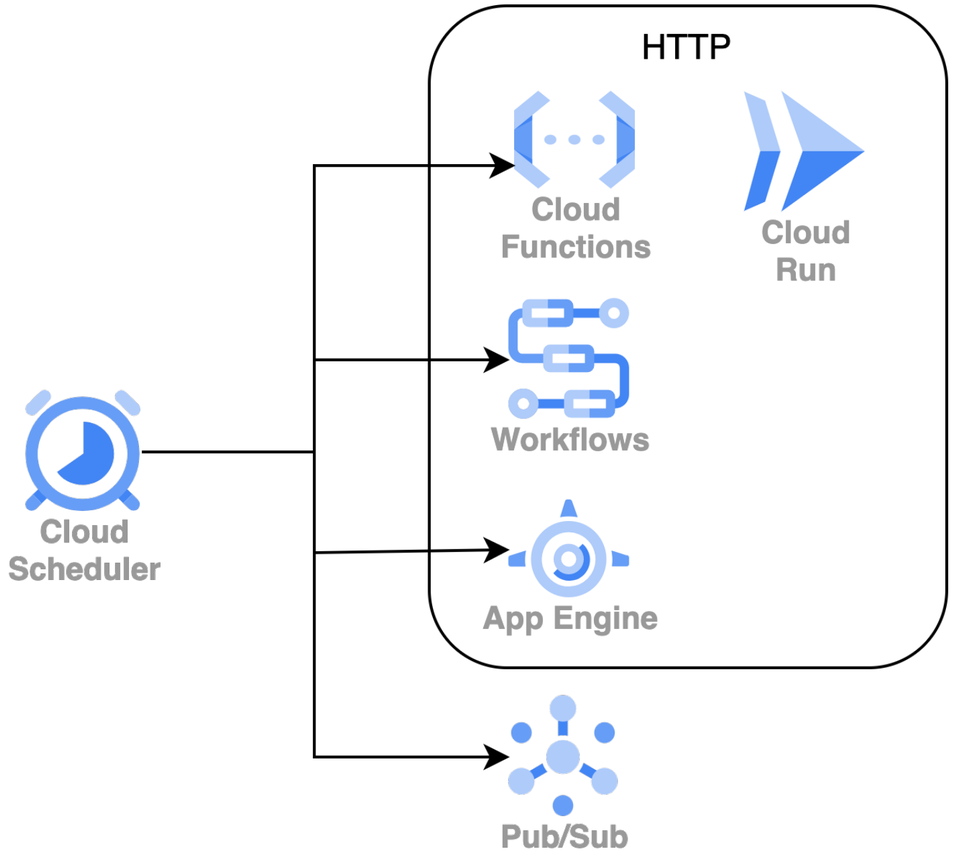
Google Cloud

Cómo orquestar en GCP

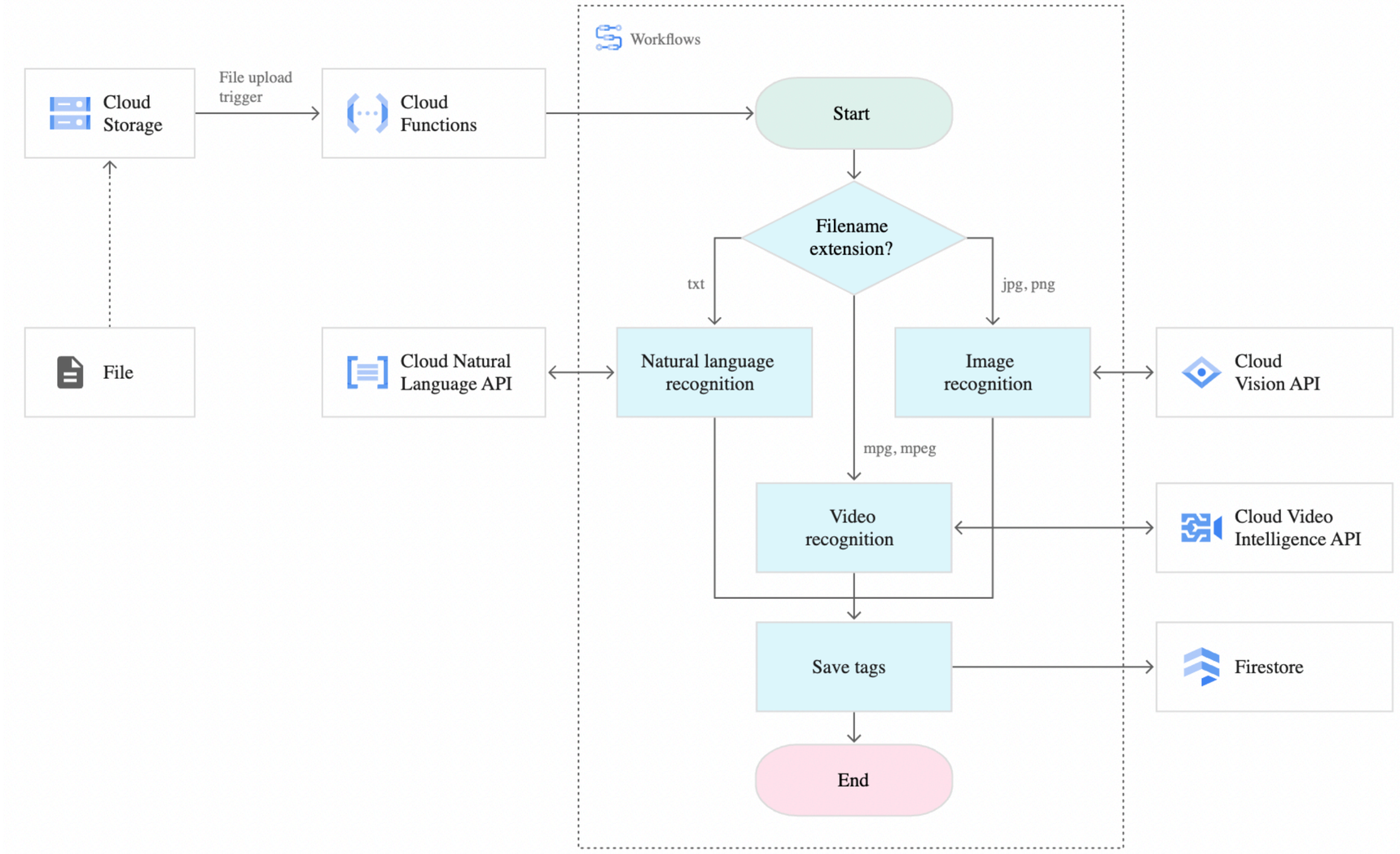
Que es Cloud Scheduler?

Cloud Scheduler es un programador de tareas totalmente administrado.



Google Cloud

Que es Workflows?



Google Cloud

Que es Cloud Composer?

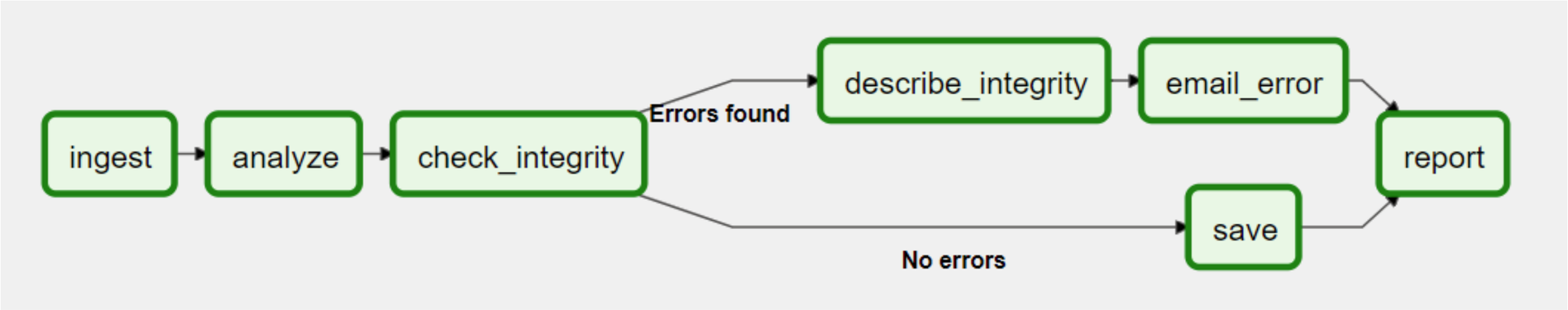
Cloud Composer es un servicio autogestionado de Apache Airflow.

Cloud Composer

Apache Airflow

python™

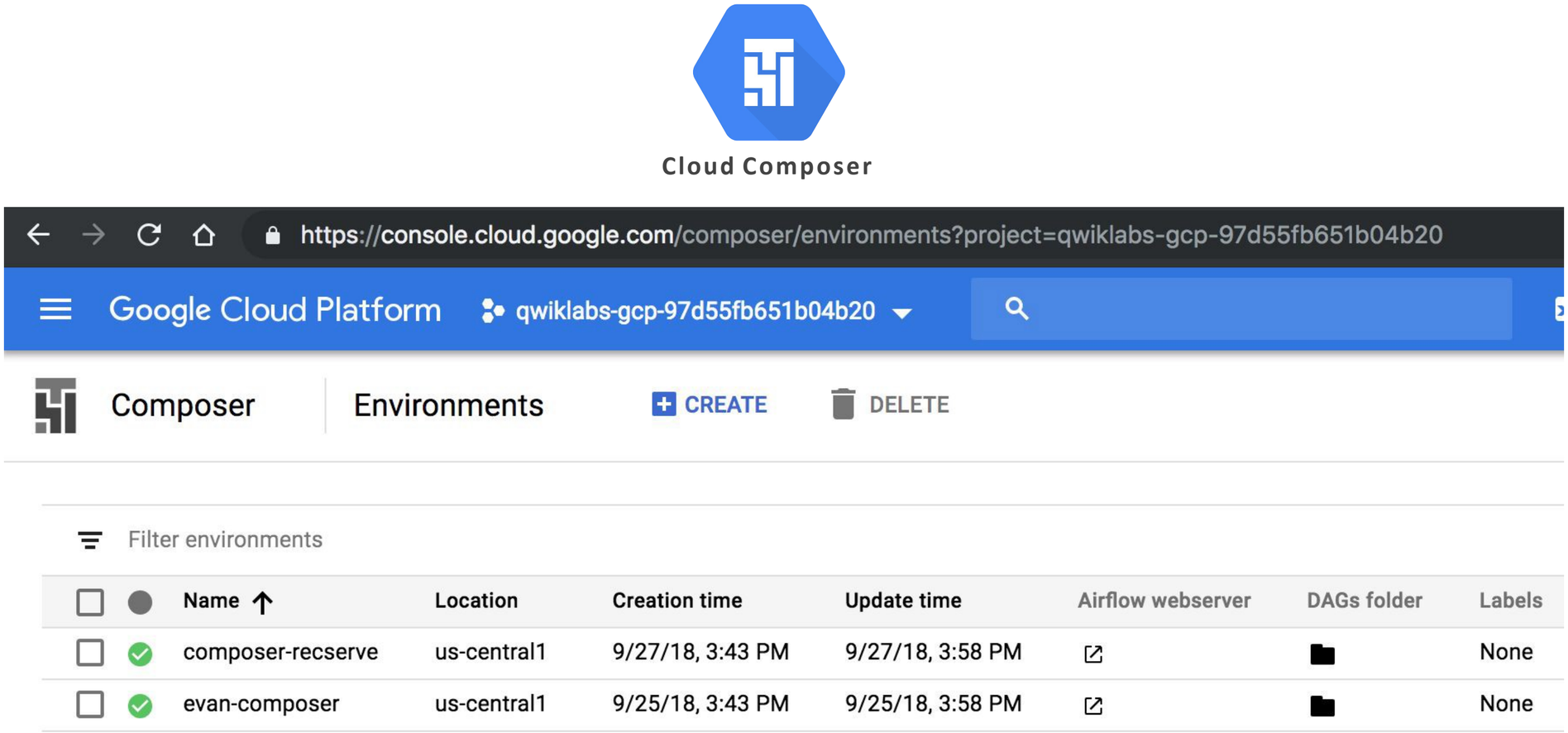
Airflow es una plataforma que le permite crear y ejecutar flujos de trabajo en python



Un flujo de trabajo se representa como un DAG (un gráfico acíclico dirigido) y contiene piezas de trabajo individuales denominadas tareas, organizadas teniendo en cuenta las dependencias y los flujos de datos.

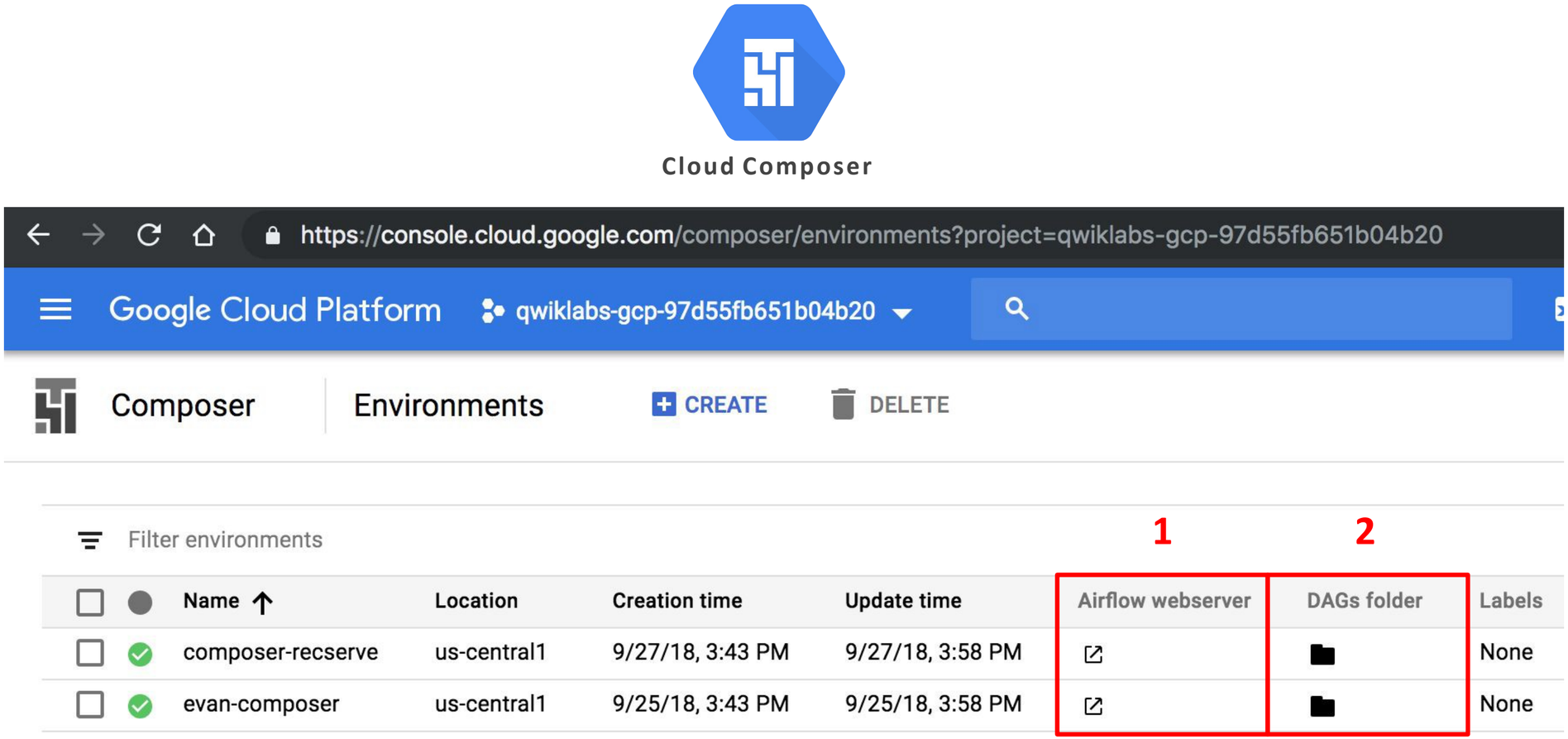
Google Cloud

Cloud Composer crea entornos administrables de Apache Airflow



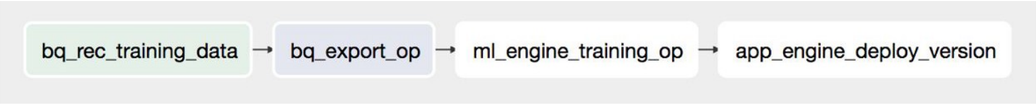
Google Cloud

Cada entorno de Airflow posee un servidor web y una carpeta separada en GCS para los DAG de canalización



Google Cloud

Ejemplo: flujo de trabajo normal de aprendizaje automático por los operadores DAG




```
# 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(t
1)
t3.set_upstream(t
2)
t4.set_upstream(t
3)
```

1

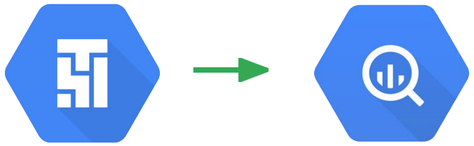
2

3

4

Google Cloud

Los operadores de GCS y BigQuery nos proporcionan información nueva de entrenamiento



```
# 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(t
1)
t3.set_upstream(t
2)
t4.set_upstream(t
3)
```

1

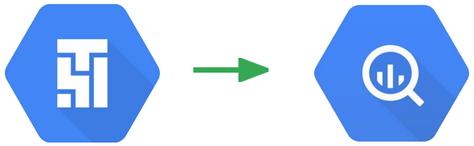
2

3

4

Google Cloud

Los operadores de GCS y BigQuery nos proporcionan información nueva de entrenamiento



```
# 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(t
1)
t3.set_upstream(t
2)
t4.set_upstream(t
3)
```

1

2

3

4

Google Cloud

Utilice el BigQueryOperator para ejecutar 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_quer

        y', 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_tabl

        e_id)
```

Google Cloud

Los comandos SQL pueden conservar parámetros (de 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',
        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_tabl

        e_id)
```

Observe las constantes de Python de un período

```
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_que

        ry', 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_tabl

        e_id)
```

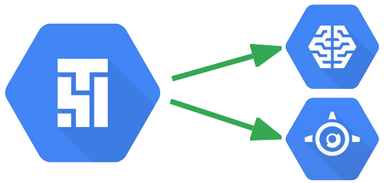
Google Cloud

Incluso puede establecer una ventana desplazable con macros

```
from airflow.contrib.operators import bigquery_operator

# constants or can be dynamic based on Airflow macros
max_query_date = '2018-02-01' # {{ macros.ds_add(ds,
-7) }} min_query_date = '2018-01-01' #
{{ macros.ds_add(ds, -1) }}
    # Query recent StackOverflow questions.
    bq_recent_questions_query =
    bigquery_operator.BigQueryOperator(
        task_id='bq_recent_questions_que

        ry', 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_tabl

        e_id)
```

Google Cloud

Los operadores de AI Platform y App Engine vuelven a implementar y entrenar nuestro modelo



```
# 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(t
1)
t3.set_upstream(t
2)
t4.set_upstream(t
3)
```

1

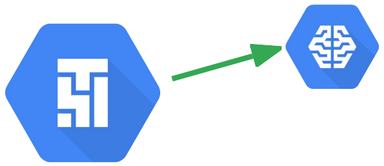
2

3

4

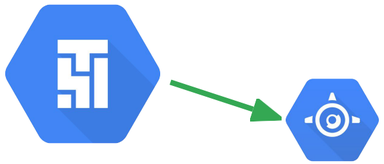
Google Cloud

Utilice los operadores de Cloud ML Engine para enviar de manera periódica nuevos trabajos de entrenamiento



```
t3 = MLEngineTrainingOperator(
    task_id='ml_engine_training_o
    p', project_id=PROJECT_ID,
    job_id=job_id,
    package_uris=[PACKAGE_URI],
    training_python_module='trainer.tas
    k', training_args=training_args,
    region=REGION,
    scale_tier='CUSTOM',
    master_type='complex_model_m_gp
    u', dag=dag
)
```

Utilice los operadores de App Engine para implementar y reimplementar los modelos de manera periódica



```
t4 = AppEngineVersionOperator(
    task_id='app_engine_deploy_versio
    n', project_id=PROJECT_ID,
    service_id='default',
    region=REGION,
    service_spec=Non
    e, dag=dag
)
```

Administre las canalizaciones y las dependencias como código

```
# 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(t
1)
t3.set_upstream(t
2)
```

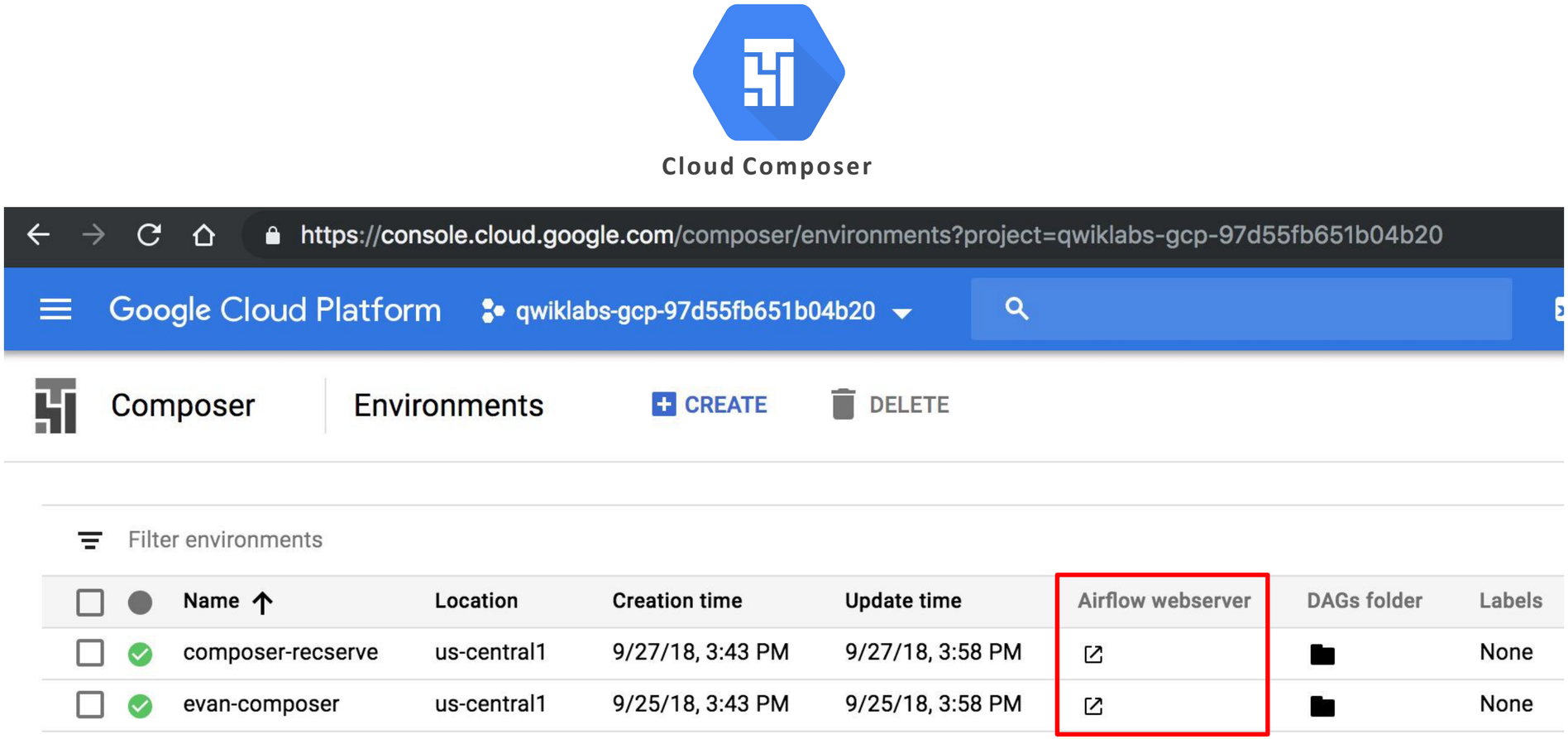
Google Cloud

Dos opciones de programación para los flujos de trabajo de Cloud Composer

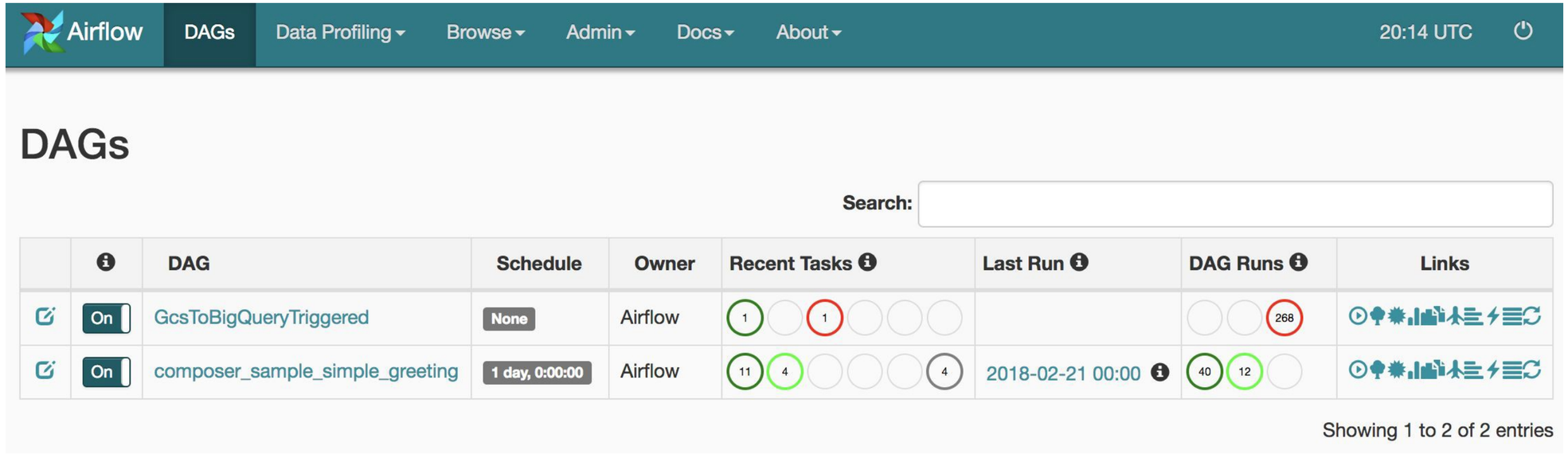
Dos opciones de programación para los flujos de trabajo de Cloud Composer:

- Periódico o
- Controlado por eventos

Inicie el servidor web de Airflow para interactuar con sus DAG

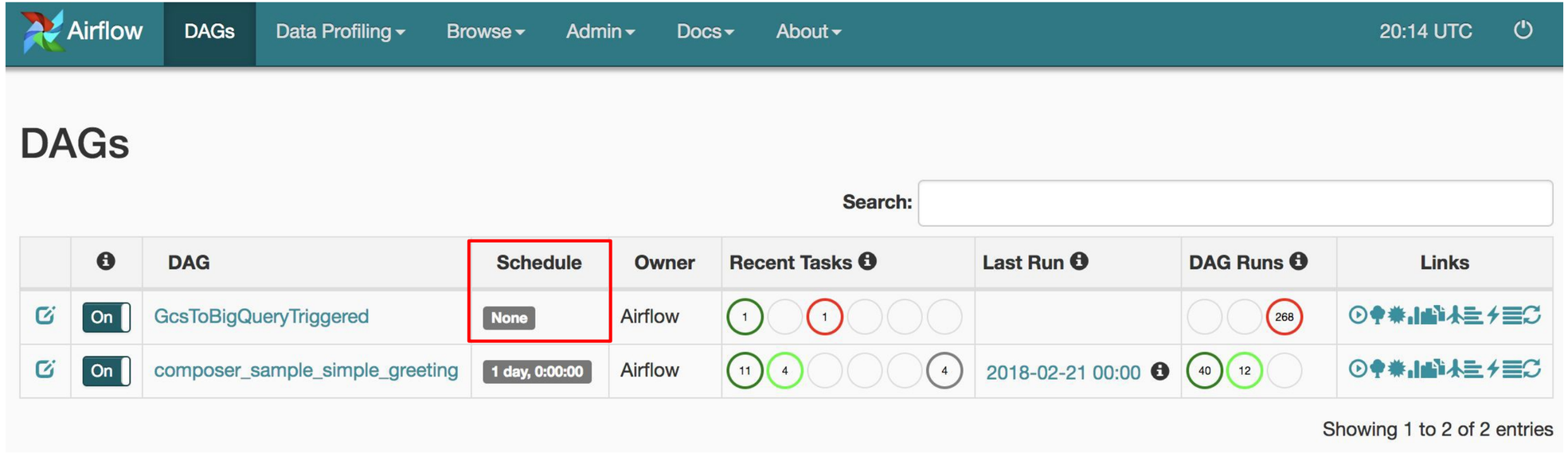


Aspectos básicos de la programación de Airflow

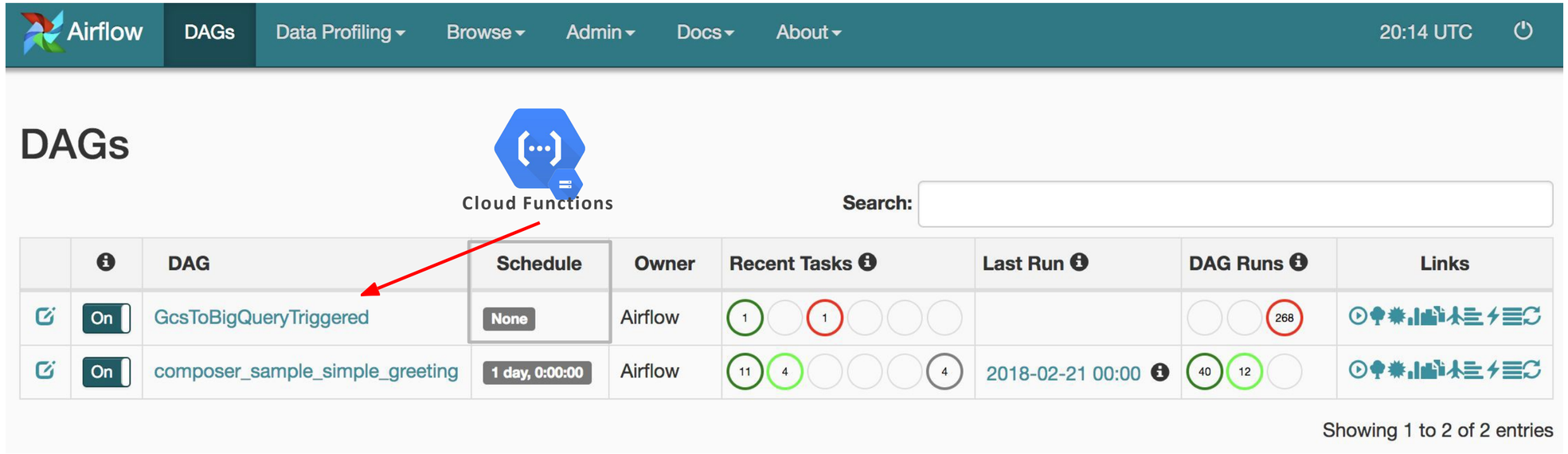


Google Cloud

¿Por qué este DAG no está programado?

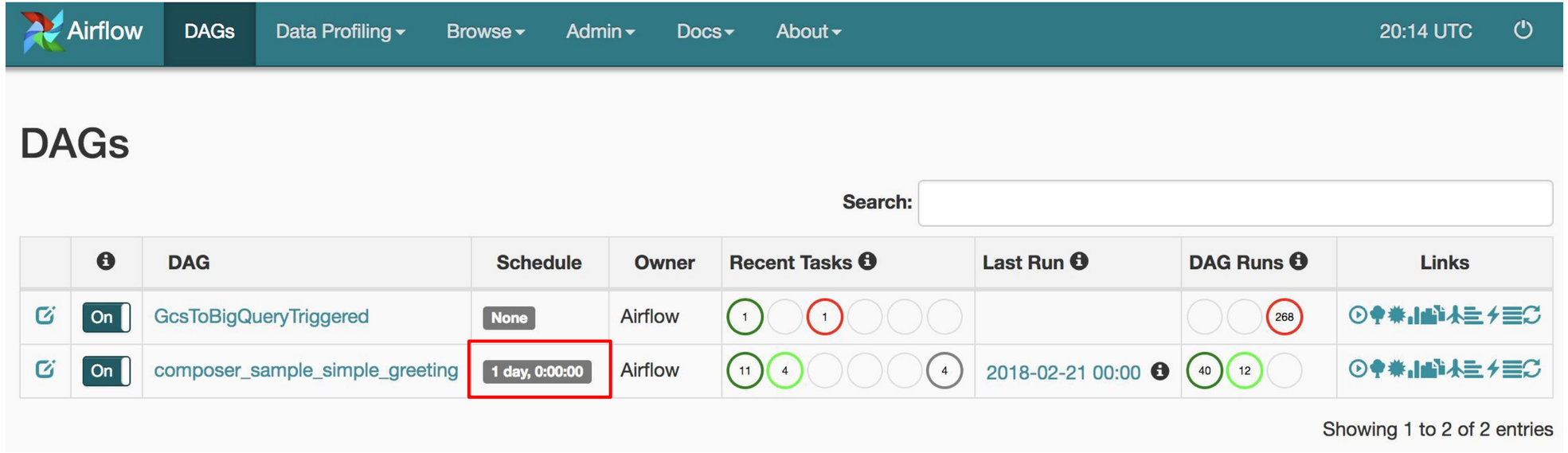


Opción 1: programación controlada por eventos con Cloud Functions



Google Cloud

Opción 2: Especifique el schedule_interval en su DAG




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

Google Cloud