

## Cloud Composer: Copying BigQuery Tables Across Different Locations

### Task 1. Create a Cloud Composer environment

1. On the Google Cloud console title bar, type **Composer** in the Search field, then click **Composer** in the Products & Page section to create a Cloud Composer environment.
2. Then click **Create environment**.
3. In dropdown menu, select **Composer 3**.
4. Set the following parameters for your environment:
  - **Name:** composer-advanced-lab
  - **Location:** us-east4
  - **Image Version:** composer-3-airflow-n.n.n-build.n *(select the highest number image available)*
  - **Service account:** Compute Engine default service account
  - Under **Environment resources**, Select **Small**.
  - Click the dropdown for **Show Advanced Configuration** and select **Airflow database zone** as us-east4-a.

Leave all other settings as default.


5. Click **Create**.

The environment creation process is completed when the green checkmark displays to

the left of the environment name on the Environments page in the Cloud Console.

**Note:** It can take up to **20 minutes** for the environment to complete the setup process. Move on to the next section `Create Cloud Storage buckets and BigQuery destination dataset`.

Click **Check my progress** to verify the objective.



Create Cloud Composer environment.

Check my progress

## Task 2. Create Cloud Storage buckets

In this task you will create two Cloud Storage Multi-Regional buckets. These buckets will be used to copy the exported tables across locations, i.e., US to EU.

### Create a bucket in US


1. Navigate to **Cloud Storage > Buckets** and click **Create**.
2. Give the bucket a universally unique name including the project ID (e.g. `qwiklabs-gcp-00-352efc2ae6b1-us`).
3. For a **Location Type** select **us (multiple regions in United States)**.

4. Leave the other value as default and click **Create**.
5. Check the box for **Enforce public access prevention on this bucket** and click **Confirm** for **Public access will be prevented** pop-up if prompted.

## Create a bucket in EU

Repeat the steps to create another bucket in **EU** region. The universally unique name should include the location as a suffix to your bucket (e.g. **qwiklabs-gcp-00-352efc2ae6b1-eu**).

Click **Check my progress** to verify the objective.



Create two Cloud Storage buckets.

**Check my progress**

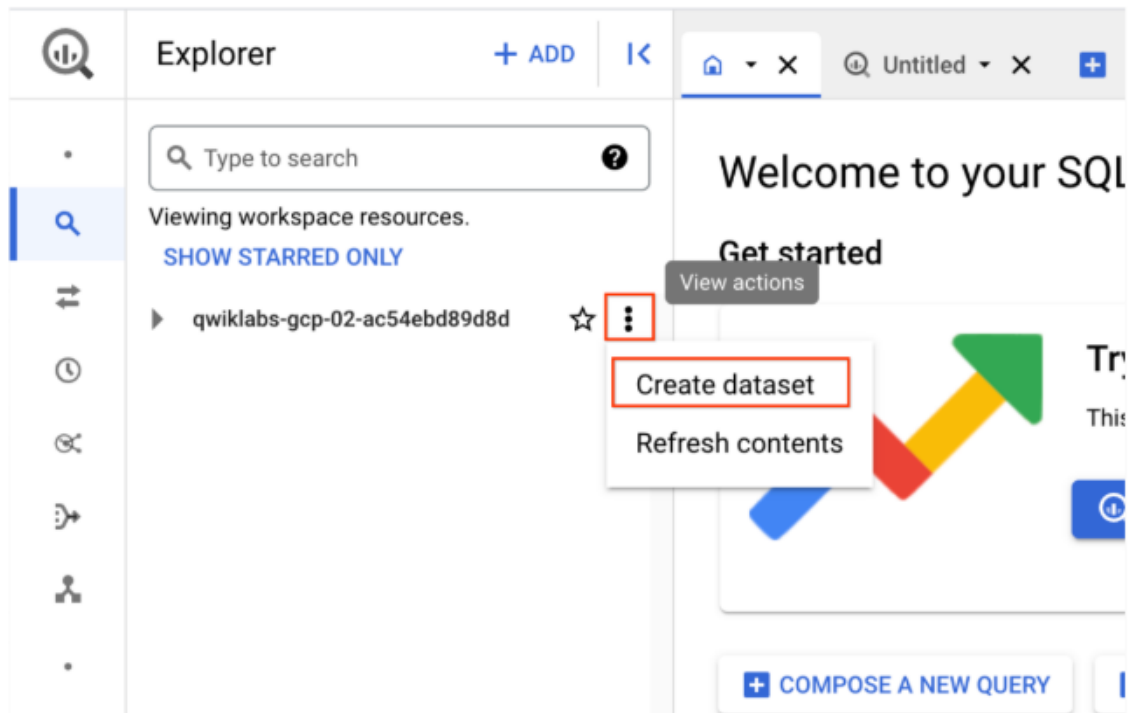
## Task 3. Create the BigQuery destination dataset

1. Create the destination BigQuery Dataset in EU from the BigQuery new web UI.
2. Go to **Navigation menu > BigQuery**.

The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and lists UI updates.

3. Click **Done**.

4. Then click the three dots next to your Qwiklabs project ID and select **Create dataset**.



5. Use the dataset ID **nyc\_tlc\_EU**, for location type choose **Multi-region** and select **EU** from the dropdown.

## Create dataset

Project ID \*

qwiklabs-gcp-04-526425f33cf9

[CHANGE](#)

Dataset ID \*

nyc\_tlc\_EU

Letters, numbers, and underscores allowed

#### Location type


☐ Region

Specify a region to colocate your datasets with other Google Cloud services.

☒ Multi-region


Allow BigQuery to select a region within a group to achieve higher quota limits.

Multi-region \*


EU (multiple regions in European Union) 

#### External Dataset

The selected region supports the following external dataset types: Cloud Spanner

☐ Link to an external dataset 

#### Default table expiration

☐ Enable table expiration 

Default maximum table age

Days

#### Tags


#### Advanced options

CREATE DATASET

CANCEL

6. Click **CREATE DATASET**.

Click **Check my progress** to verify the objective.



Create a dataset.

**Check my progress**

## Task 4. Airflow and core concepts, a brief introduction

- While your environment is building, read about the sample file you'll be using in this lab.

[Airflow](#) is a platform to programmatically author, schedule and monitor workflows.

Use airflow to author workflows as directed acyclic graphs (DAGs) of tasks. The airflow scheduler executes your tasks on an array of workers while following the specified dependencies.

### Core concepts

[DAG](#) - A Directed Acyclic Graph is a collection of tasks, organized to reflect their relationships and dependencies.

[Operator](#) - The description of a single task, it is usually atomic. For example, the *BashOperator* is used to execute bash command.

[Task](#) - A parameterised instance of an Operator; a node in the DAG.

[Task Instance](#) - A specific run of a task; characterized as: a DAG, a Task, and a point in time. It has an indicative state: *running, success, failed, skipped, ...*

Learn more about Airflow concepts from the [Concepts documentation](#).

## Task 5. Define the workflow

Cloud Composer workflows are comprised of [DAGs \(Directed Acyclic Graphs\)](#). The code shown in [bq\\_copy\\_across\\_locations.py](#) is the workflow code, also referred to as the DAG. Open the file now to see how it is built. Next will be a detailed look at some of the key components of the file.

To orchestrate all the workflow tasks, the DAG imports the following operators:

1. `DummyOperator` : Creates Start and End dummy tasks for better visual representation of the DAG.
  2. `BigQueryToCloudStorageOperator` : Exports BigQuery tables to Cloud Storage buckets using Avro format.
  3. `GoogleCloudStorageToGoogleCloudStorageOperator` : Copies files across Cloud Storage buckets.
  4. `GoogleCloudStorageToBigQueryOperator` : Imports tables from Avro files in Cloud Storage bucket.
- In this example, the function `read_table_list()` is defined to read the config file and build the list of tables to copy:

```
# -----  
# Functions  
# -----
```

```
def read_table_list(table_list_file):
```

```
    """
```

```
    Reads the master CSV file that will help in creating Airflow tasks in
```

the DAG dynamically.

:param table\_list\_file: (String) The file location of the master file,

e.g. '/home/airflow/framework/master.csv'

:return master\_record\_all: (List) List of Python dictionaries containing

the information for a single row in master CSV file.

"""

```
master_record_all = []
```

```
logger.info('Reading table_list_file from : %s' % str(table_list_file))
```

```
try:
```

```
    with open(table_list_file, 'rb') as csv_file:
```

```
        csv_reader = csv.reader(csv_file)
```

```
        next(csv_reader) # skip the headers
```

```
        for row in csv_reader:
```

```
            logger.info(row)
```

```
            master_record = {
```

```
                'table_source': row[0],
```

```
                'table_dest': row[1]
```

```
            }
```

```
            master_record_all.append(master_record)
```

```
        return master_record_all
```

```
except IOError as e:
```

```
    logger.error('Error opening table_list_file %s: ' % str(
```

```
        table_list_file), e)
```

- The name of the DAG is bq\_copy\_us\_to\_eu\_01, and the DAG is not scheduled by default so needs to be triggered manually.

```
default_args = {
```

```
    'owner': 'airflow',
```

```
    'start_date': datetime.today(),
```

```
    'depends_on_past': False,
```

```
    'email': [''],
```



```
'email_on_failure': False,
'email_on_retry': False,
'retries': 1,
'retry_delay': timedelta(minutes=5),
}
# DAG object.
with models.DAG('bq_copy_us_to_eu_01',
                default_args=default_args,
                schedule_interval=None) as dag:
```

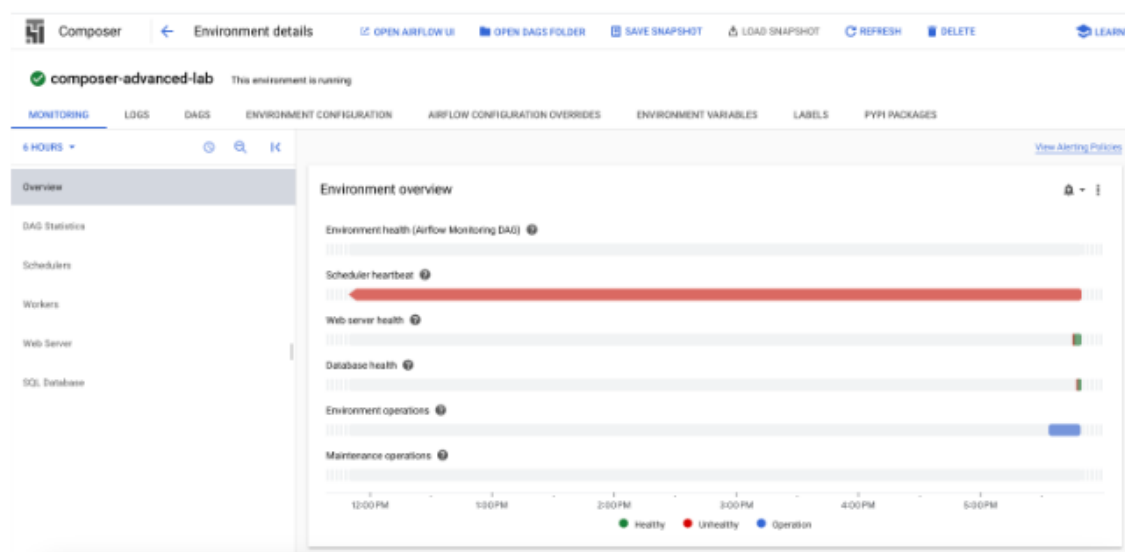
- To define the Cloud Storage plugin, the class CloudStoragePlugin(AirflowPlugin) is defined, mapping the hook and operator downloaded from the Airflow 1.10-stable branch.

```
# Import operator from plugins
from gcs_plugin.operators import gcs_to_gcs
```

## Task 6. View environment information

1. Go back to **Composer** to check on the status of your environment.
2. Once your environment has been created, click the name of the environment to see its details.

The **Environment details** page provides information, such as the Airflow web UI URL, Google Kubernetes Engine cluster ID, name of the Cloud Storage bucket connected to the DAGs folder.



**Note:** Cloud Composer uses [Cloud Storage](#) to store Apache Airflow DAGs, also known as *workflows*. Each environment has an associated Cloud Storage bucket. Cloud Composer schedules only the DAGs in the Cloud Storage bucket.

The next steps should be completed in Cloud Shell.

## Creating a virtual environment

Python virtual environments are used to isolate package installation from the system.

1. Install the `virtualenv` environment:

```
sudo apt-get install -y virtualenv
```



2. Build the virtual environment:

```
python3 -m venv venv
```



3. Activate the virtual environment:

```
source venv/bin/activate
```



## Task 7. Create a variable for the DAGs Cloud Storage bucket

- In Cloud Shell, run the following to copy the name of the DAGs bucket from your Environment Details page and set a variable to refer to it in Cloud Shell:

**Note:** Make sure to replace your DAGs bucket name in the following command. Navigate to **Navigation menu > Cloud Storage**, it will be similar to `us-east4-composer-advanced-YOURDAGSBUCKET-bucket`.

```
DAGS_BUCKET=<your DAGs bucket name>
```



You will be using this variable a few times during the lab.

## Task 8. Set Airflow variables

Airflow variables are an Airflow-specific concept that is distinct from [environment variables](#). In this step, you'll set the following three [Airflow variables](#) used by the DAG we will deploy: `table_list_file_path`, `gcs_source_bucket`, and `gcs_dest_bucket`.

Key	Value	Details
<code>table_list_file_path</code>	<code>/home/airflow/gcs/dags/bq_copy_eu_to_us_sample.csv</code>	CSV file listing source and target tables, including dataset

<code>gcs_source_bucket</code>	{UNIQUE ID}-us	Cloud Storage bucket to use for exporting BigQuery tables from source
<code>gcs_dest_bucket</code>	{UNIQUE ID}-eu	Cloud Storage bucket to use for importing BigQuery tables at destination

The next `gcloud composer` command executes the Airflow CLI sub-command [variables](#). The sub-command passes the arguments to the `gcloud` command line tool.

To set the three variables, you will run the `composer command` once for each row from the above table. The form of the command is this:

```
gcloud composer environments run ENVIRONMENT_NAME \
--location LOCATION variables -- \
set KEY VALUE
```



You can safely ignore this gcloud error: (ERROR: gcloud crashed (TypeError): 'NoneType' object is not callable). This is a [known issue](#) with using gcloud composer environments run with the 410.0.0 version of gcloud. Your variables will still be set accordingly despite the error message.

- `ENVIRONMENT_NAME` is the name of the environment.
- `LOCATION` is the Compute Engine region where the environment is located. The gcloud composer command requires including the `--location` flag or [setting the default location](#) before running the gcloud command.
- `KEY` and `VALUE` specify the variable and its value to set. Include a space two dashes space ( `--` ) between the left-side gcloud command with gcloud-related arguments and the right-side Airflow sub-command-related arguments. Also include a space between the `KEY` and `VALUE` arguments. using the `gcloud composer environments run` command with the variables sub-command in

Run these commands in Cloud Shell, replacing `gcs_source_bucket` and `gcs_dest_bucket` by the names of the buckets you created on Task 2.

```
gcloud composer environments run composer-advanced-lab \
--location us-east4 variables -- \
set table_list_file_path
/home/airflow/gcs/dags/bq_copy_eu_to_us_sample.csv

gcloud composer environments run composer-advanced-lab \
--location us-east4 variables -- \
set gcs_source_bucket {UNIQUE ID}-us

gcloud composer environments run composer-advanced-lab \
--location us-east4 variables -- \
set gcs_dest_bucket {UNIQUE_ID}-eu
```

To see the value of a variable, run the Airflow CLI sub-command [variables](#) with the `get` argument or use the [Airflow UI](#).

For example, run the following:

```
gcloud composer environments run composer-advanced-lab \
--location us-east4 variables -- \
get gcs_source_bucket
```

**Note:** Make sure to set all three Airflow variables used by the DAG.

## Task 9. Upload the DAG and dependencies to Cloud Storage

1. Copy the Google Cloud Python docs samples files into your Cloud shell:

```
cd ~  
gcloud storage cp -r gs://spls/gsp283/python-docs-samples .
```



2. Upload a copy of the third party hook and operator to the plugins folder of your Composer DAGs Cloud Storage bucket:

```
gcloud storage cp -r python-docs-samples/third_party/apache-airflow/plugins/* gs://$DAGS_BUCKET/plugins
```



3. Next, upload the DAG and config file to the DAGs Cloud Storage bucket of your environment:

```
gcloud storage cp python-docs-samples/composer/workflows/bq_copy_across_locations.py gs://$DAGS_BUCKET/dags  
gcloud storage cp python-docs-samples/composer/workflows/bq_copy_eu_to_us_sample.csv gs://$DAGS_BUCKET/dags
```



Cloud Composer registers the DAG in your Airflow environment automatically, and DAG changes occur within 3-5 minutes. You can see task status in the Airflow web interface and confirm the DAG is not scheduled as per the settings.

## Task 10. Explore the Airflow UI

To access the Airflow web interface using the Cloud Console:

1. Go back to the Composer **Environments** page.
2. In the **Airflow webserver** column for the environment, click the **Airflow** link.

Composer

Environments

CREATE

DELETE

Filter

Filter environments

<input type="checkbox"/>	<div><div></div><div>Name</div><div>↑</div></div>	Location	Creation time	Update time	Airflow webserver	Logs	DAGs folder	Labels
<input type="checkbox"/>	<div><div></div><div>composer-advanced-lab</div></div>	us-central1	5/12/21, 7:38 PM	5/12/21, 7:54 PM	<div><div></div><div>Airflow</div></div>	<div><div></div><div>Logs</div></div>	<div><div></div><div>DAGs</div></div>	None

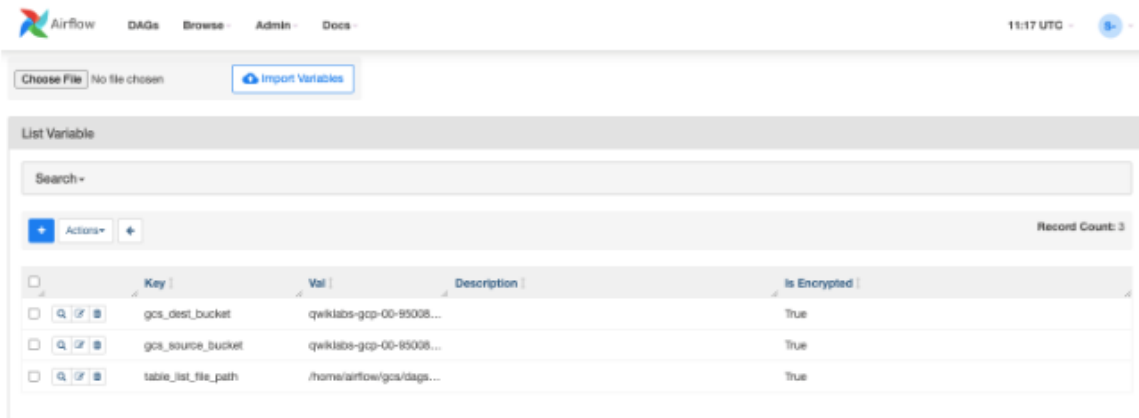
3. Click on your lab credentials.
4. The Airflow web UI opens in a new browser window. Data will still be loading when you get here. You can continue with the lab while this is happening.

## Viewing variables

The variables you set earlier are persisted in your environment.

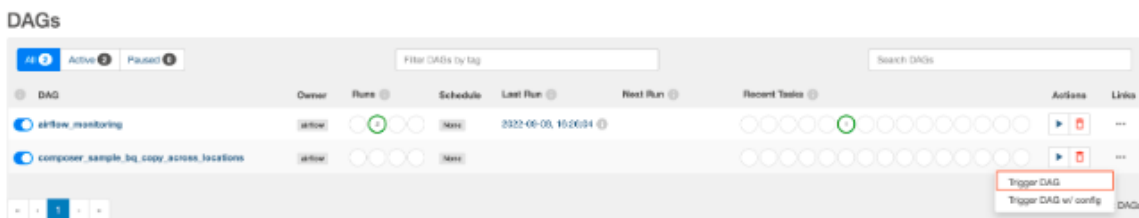
- View the variables by selecting **Admin > Variables** from the Airflow menu bar.





## Trigger the DAG to run manually

1. Click on the **DAGs** tab and wait for the links to finish loading.
2. To trigger the DAG manually, click the play button for `composer_sample_bq_copy_across_locations` :



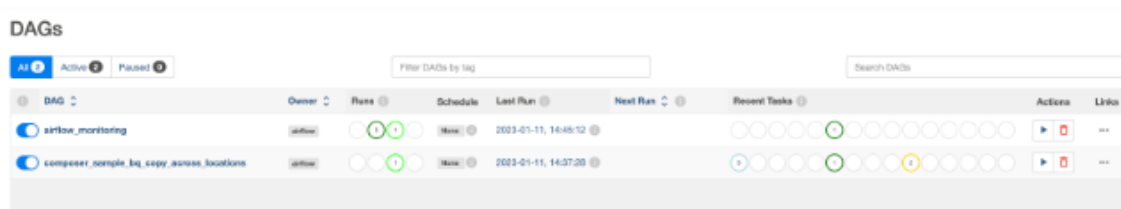
3. Click **Trigger DAG** to confirm this action.

Click **Check my progress** to verify the objective.

## Exploring DAG runs

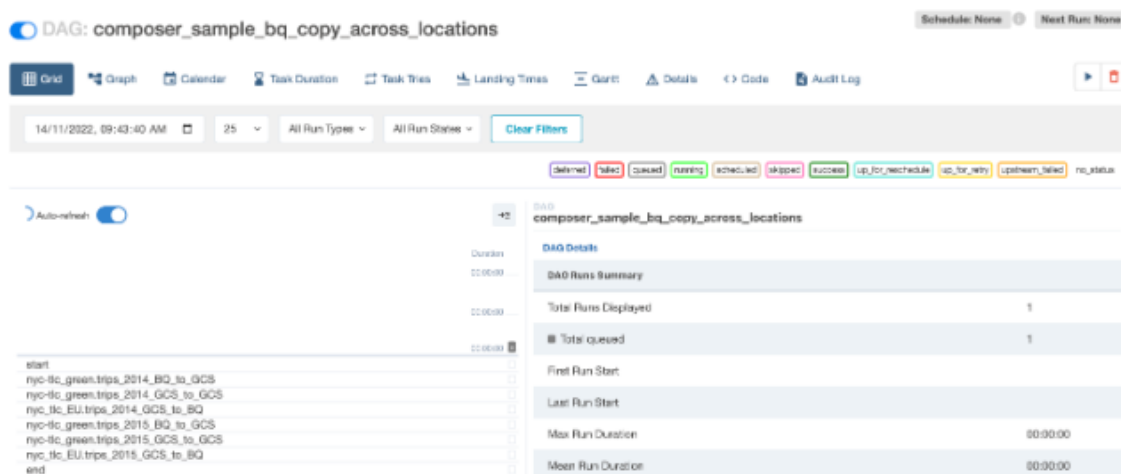
When you upload your DAG file to the DAGs folder in Cloud Storage, Cloud Composer parses the file. If no errors are found, the name of the workflow appears in the DAG listing, and the workflow is queued to run immediately if the schedule conditions are met, in this case, None as per the settings.

The **Runs** status turns green once the play button is pressed:



DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
airflow_monitoring	airflow		None	2023-01-11, 14:45:12				...
composer_sample_bq_copy_across_locations	airflow		None	2023-01-11, 14:37:28				...

1. Click the name of the DAG to open the DAG details page. This page includes a graphical representation of workflow tasks and dependencies.



**DAG: composer\_sample\_bq\_copy\_across\_locations** Schedule: None Next Run: None

Grid Graph Calendar Task Duration Task Times Landing Times Gantt Details Code Audit Log

14/11/2022, 09:43:40 AM 25 All Run Types All Run States Clear Filters

defined failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh

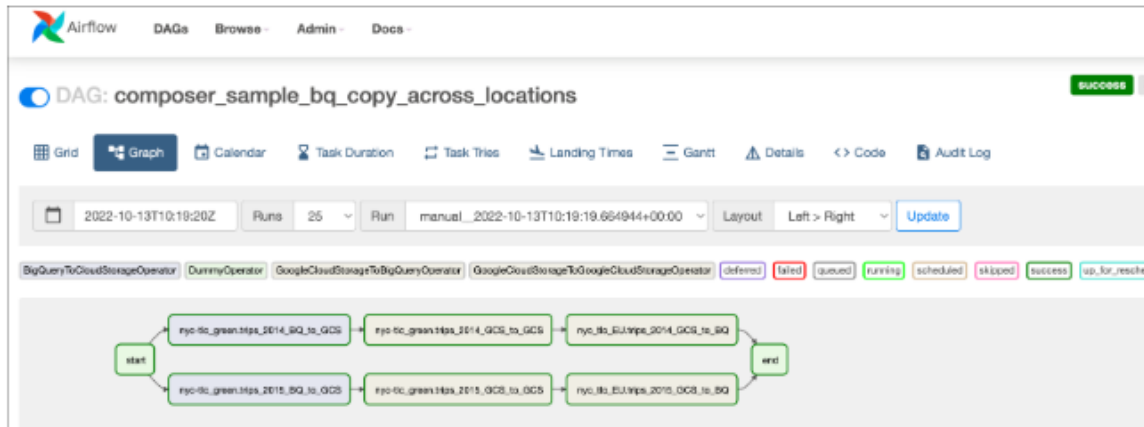
**composer\_sample\_bq\_copy\_across\_locations**

**DAG Details**

**DAG Runs Summary**

Total Runs Displayed	1
Total queued	1
First Run Start	
Last Run Start	
Max Run Duration	00:00:00
Mean Run Duration	00:00:00

2. Now, in the toolbar, click **Graph**, then mouseover the graphic for each task to see its status. Note that the border around each task also indicates the status (green border = running; red = failed, etc.).



To run the workflow again from the **Graph** view:

1. In the Airflow UI Graph View, click the **start** graphic.
2. Click **Clear** to reset all the tasks and then click **OK** to confirm.

Task Instance: **start**  
at: 2022-09-08, 16:30:04 UTC

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

Download Log (by attempts):

### 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

Refresh your browser while the process is running to see the most recent information.

## Task 11. Validate the results

Now check the status and results of the workflow by going to these Cloud Console pages:

- The exported tables were copied from the US bucket to the EU Cloud Storage bucket. Click on **Cloud Storage** to see the intermediate Avro files in the source (US) and destination (EU) buckets.
- The list of tables were imported into the target BigQuery Dataset. Click on **BigQuery**, then click on your project name and the **nyc\_tlc\_EU** dataset to validate the tables are accessible from the dataset you created.

## Delete the Cloud Composer environment

1. Return to the **Environments** page in Cloud Composer.

2. Select your Cloud Composer environment.
3. Click **Delete**.
4. In the dialog that opens, click **Delete** to confirm you want to continue deleting the Cloud Composer environment.

## Congratulations!

You copied tables programmatically from US to EU! This lab is based on this [blog post](#) by David Sabater Dinter.

## Next steps

- Learn more about using Airflow at the [Airflow website](#) or the Airflow [GitHub project](#).
- There are lots of other resources available for Airflow, including a [discussion group](#).
- Sign up for an Apache JIRA account and re-open any issues that you care about in the [Apache Airflow JIRA project](#).
- For information about the Airflow UI, see [Accessing the web interface](#).