

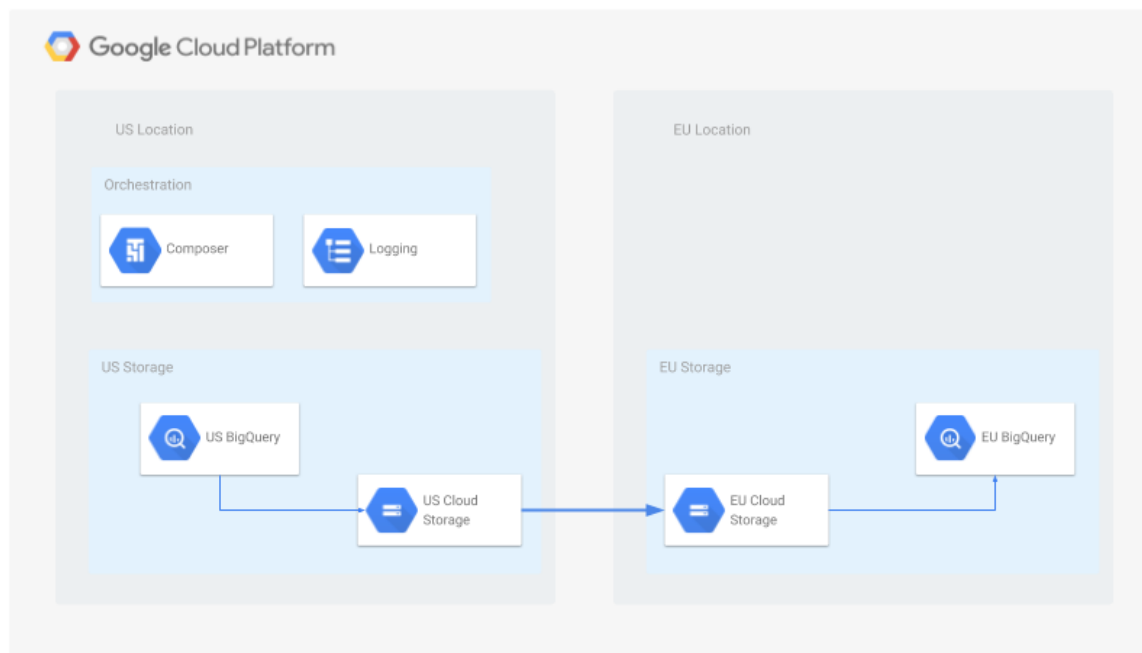
# Cloud Composer: Copying BigQuery Tables Across Different Locations

## Overview

In this advanced lab, you will learn how to create and run an [Apache Airflow](#) workflow in Cloud Composer that completes the following tasks:

- Reads from a config file the list of tables to copy
- Exports the list of tables from a [BigQuery](#) dataset located in US to [Cloud Storage](#)
- Copies the exported tables from US to EU [Cloud Storage](#) buckets
- Imports the list of tables into the target BigQuery Dataset in EU

Copy BigQuery tables across locations



## Create Cloud Composer environment

First, create a Cloud Composer environment by clicking on **Composer** in the **Navigation menu**.

Then click **Create environment**. In dropdown menu, select **Composer 1**

Set the following parameters for your environment:

- **Name:** composer-advanced-lab
- **Location:** us-central1
- **Zone:** us-central1-a

Leave all other settings as default. 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.

*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.*

## Create Cloud Storage buckets

Create two Cloud Storage Multi-Regional buckets. Give your two buckets a universally unique name including the location as a suffix:

- one located in the US as *source* (e.g. 6552634-us)
- the other located in EU as *destination* (e.g. 6552634-eu)

These buckets will be used to copy the exported tables across locations, i.e., US to EU.

## BigQuery destination dataset

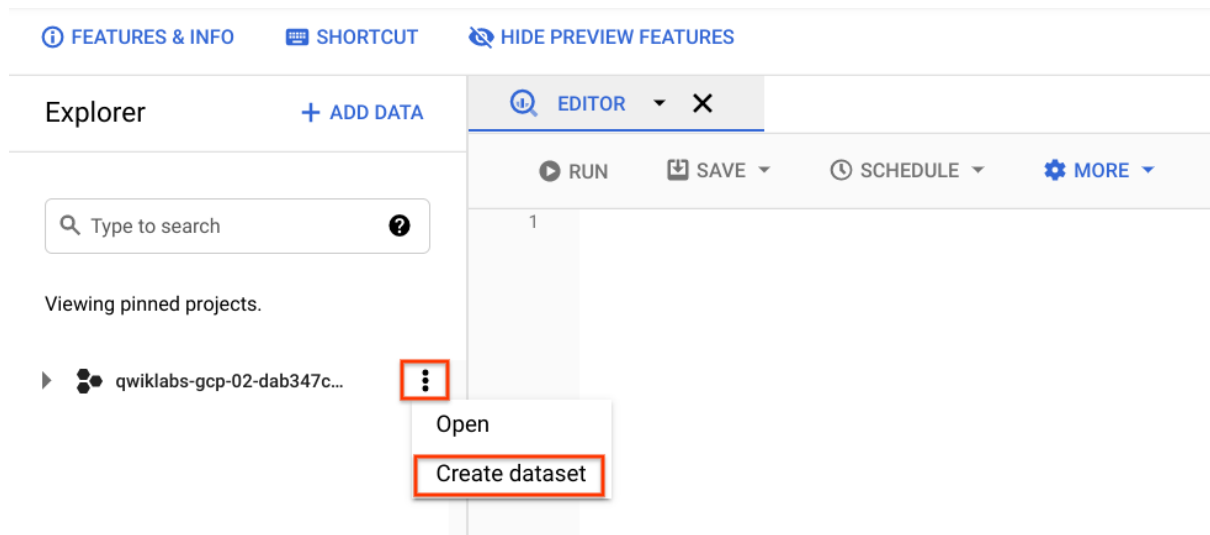
Create the destination BigQuery Dataset in EU from the BigQuery new web UI.

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.

Click Done.

Then click the three dots next to your Qwiklabs project ID and select Create dataset:



Use the name `nyc_tlc_EU` and Data location EU.

### Create dataset

**Dataset ID \***

`nyc_tlc_EU`

Letters, numbers, and underscores allowed

**Data location**

`eu` (multiple regions in European Union)

### Default table expiration

☐ Enable table expiration ?

Default maximum table age

Days

### Encryption

☒ Google-managed encryption key

No configuration required

☐ Customer-managed encryption key (CMEK)

Manage via Google Cloud Key Management Service

**CREATE DATASET**

**CANCEL**

Click **CREATE DATASET**.

## 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, organised 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 commands.
- [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, ...*
- The rest of the Airflow concepts can be found [here](#).

### Defining 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_master_file()` 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.

```
"""
    GCS Plugin
    This plugin provides an interface to GCS operator from Airflow
    Master.
"""
from airflow.plugins_manager import AirflowPlugin
from gcs_plugin.hooks.gcs_hook import GoogleCloudStorageHook
from gcs_plugin.operators.gcs_to_gcs import \
    GoogleCloudStorageToGoogleCloudStorageOperator
class GCSPlugin(AirflowPlugin):
    name = "gcs_plugin"
    operators = [GoogleCloudStorageToGoogleCloudStorageOperator]
    hooks = [GoogleCloudStorageHook]
```

## Viewing environment information

Go back to **Composer** to check on the status of your environment.

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.

Google Cloud Platform

qwklabs-gcp-02-7000ba500708

Composer

Environment details

REFRESH

DELETE

composer-advanced-lab

This environment is running

MONITORING

BETA

ENVIRONMENT CONFIGURATION

AIRFLOW CONFIGURATION OVERRIDES

ENVIRONMENT VARIABLES

LABELS

PYPI PACKAGES

EDIT

UPGRADE IMAGE VERSION

BETA

Name	composer-advanced-lab		
Zone	us-central1-a		
Service account	133321383364-compute@developer.gserviceaccount.com		
Google API scopes	https://www.googleapis.com/auth/cloud-platform		
GKE cluster	projects/qwklabs-gcp-02-7000ba500708/zones/us-central1-a/clusters/us-central1-composer-advanc-6f44c1aa-gke		
	Details	<a href="#">view cluster details</a>	
	Workloads	<a href="#">view cluster workloads</a>	
Image version	composer-1.11.3-airflow-1.10.6		
Python version	3		
Network tags	None		
Worker nodes			
	Node count	3	
	Disk size (GB)	100	
	Machine type	n1-standard-1	
Cloud SQL configuration			
	Machine type	db-n1-standard-2 (2 vCPU, 7.5 GB memory) <a href="#">EDIT</a>	
Network configuration			
	VPC-native	Disabled	
	Network ID	projects/qwklabs-gcp-02-7000ba500708/global/networks/default	
	Subnetwork ID	-	
Private environment	Disabled		
Web server configuration			
	Network access control	All IP addresses have access (default) <a href="#">EDIT</a>	
	Machine type	composer-n1-webserver-2 (2 vCPU, 1.6 GB memory) <a href="#">EDIT</a>	
DAGs folder	<a href="#">gs://us-central1-composer-advanc-6f44c1aa-bucket/dags</a>		
Airflow web UI	<a href="#">https://4de945cdc82a31f5p.tp.appspot.com</a>		
Stackdriver	<a href="#">view logs</a>		
Created	Wed Sep 09 2020 22:35:47 GMT+0530 (India Standard Time)		
Updated	Wed Sep 09 2020 22:52:49 GMT+0530 (India Standard Time)		

**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.

## Creating a virtual environment

Execute the following command to download and update the packages list.

```
sudo apt-get update
```

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

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

If prompted [Y/n], press Y and then Enter.

```
virtualenv -p python3 venv
```

Activate the virtual environment.

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

## Setting 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:

*Make sure to replace your DAGs bucket name in the following command. Navigate to Navigation menu > Storage, it will be similar to*  
*us-central1-composer-advanc-YOURDAGSBUCKET-bucket.*

```
DAGS_BUCKET=us-central1-composer-advanc-YOURDAGSBUCKET-bucket
```

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

## Setting 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>	<code>{UNIQUE ID}-us</code>	Cloud Storage bucket to use for exporting BigQuery tabledest_bbucks from source



gcs_dest_bucket	{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
```

- `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

For example, the `gcs_source_bucket` variable would be set like this:

```
gcloud composer environments run composer-advanced-lab \
--location us-central1 variables -- \
--set gcs_source_bucket My_Bucket-us
```

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-central1 variables -- \
  --get gcs_source_bucket
```

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

## Uploading the DAG and dependencies to Cloud Storage

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

```
cd ~
```

```
gsutil -m 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:

```
gsutil 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:

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

## Using 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

<div><input type="checkbox"/></div> <div><input checked="" type="radio"/></div>	Name <div>↑</div>	Location	Creation time	Update time	Airflow webserver	Logs	DAGs folder	Labels
<div><input type="checkbox"/></div> <div><input checked="" type="radio"/></div>	composer-advanced-lab	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. You can view the variables by selecting **Admin > Variables** from the Airflow menu bar.

Airflow		DAGs	Data Profiling ▾	Browse ▾	Admin ▾	Docs ▾	About ▾
Variables							
Choose File No file chosen		Import Variables					
List (3)	Create	Add Filter ▾	With selected ▾	Search: key, val, is_encrypted			
<input type="checkbox"/>		Key	Val				
<input type="checkbox"/>	 	gcs_dest_bucket	ex-eu				
<input type="checkbox"/>	 	gcs_source_bucket	ex-us				
<input type="checkbox"/>	 	table_list_file_path	/home/airflow/gcs/dags/bq_copy_eu_to_us_sample.csv				

## Trigger the DAG to run manually

Click on the **DAGs** tab and wait for the links to finish loading.

To trigger the DAG manually, click the play button for composer\_sample\_bq\_copy\_across\_locations :

DAG Runs ⓘ	Links									
2										

Click **Trigger** to confirm this action.

## 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.

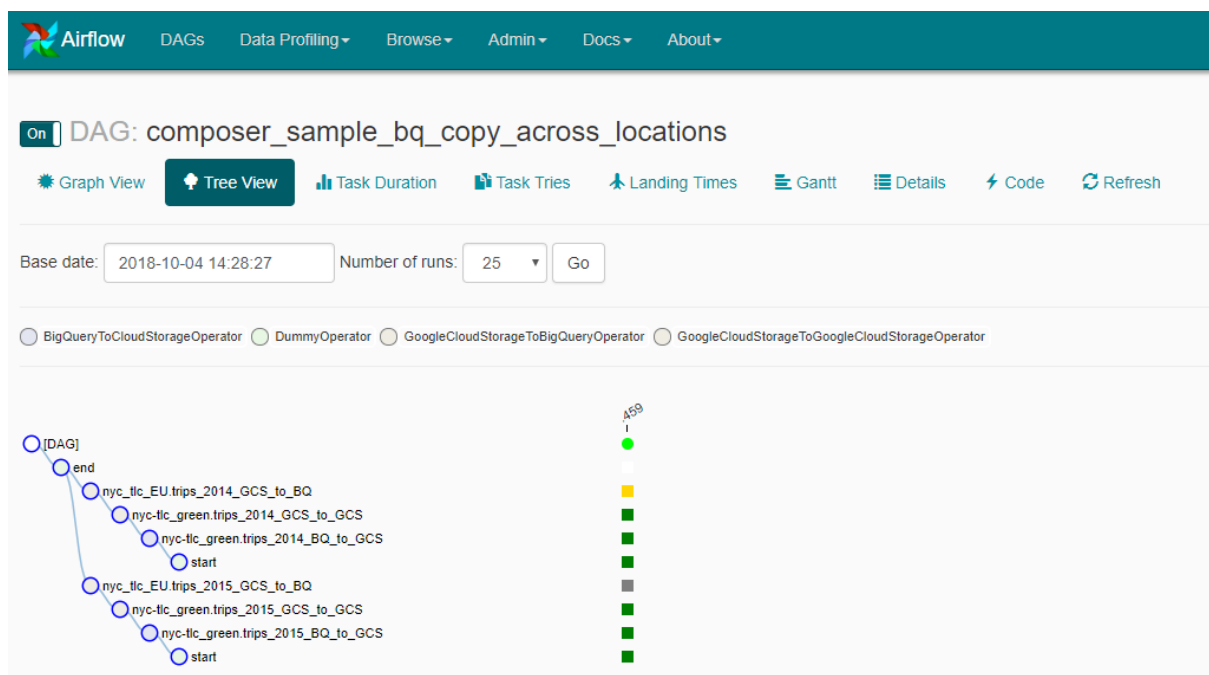
The **DAG Runs** status turns green once the play button is pressed:

DAGs

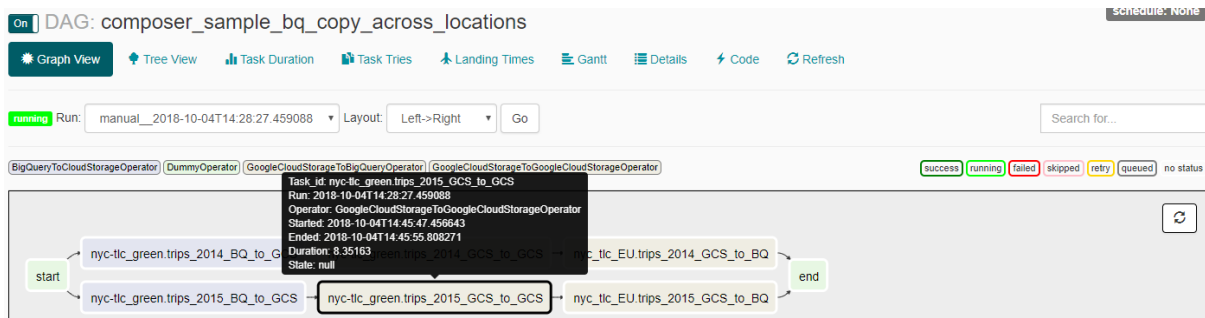
Search: <input type="text"/>							
	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	<code>composer_sample_bq_copy_across_locations</code>	<code>None</code>	airflow				

Showing 1 to 1 of 1 entries

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

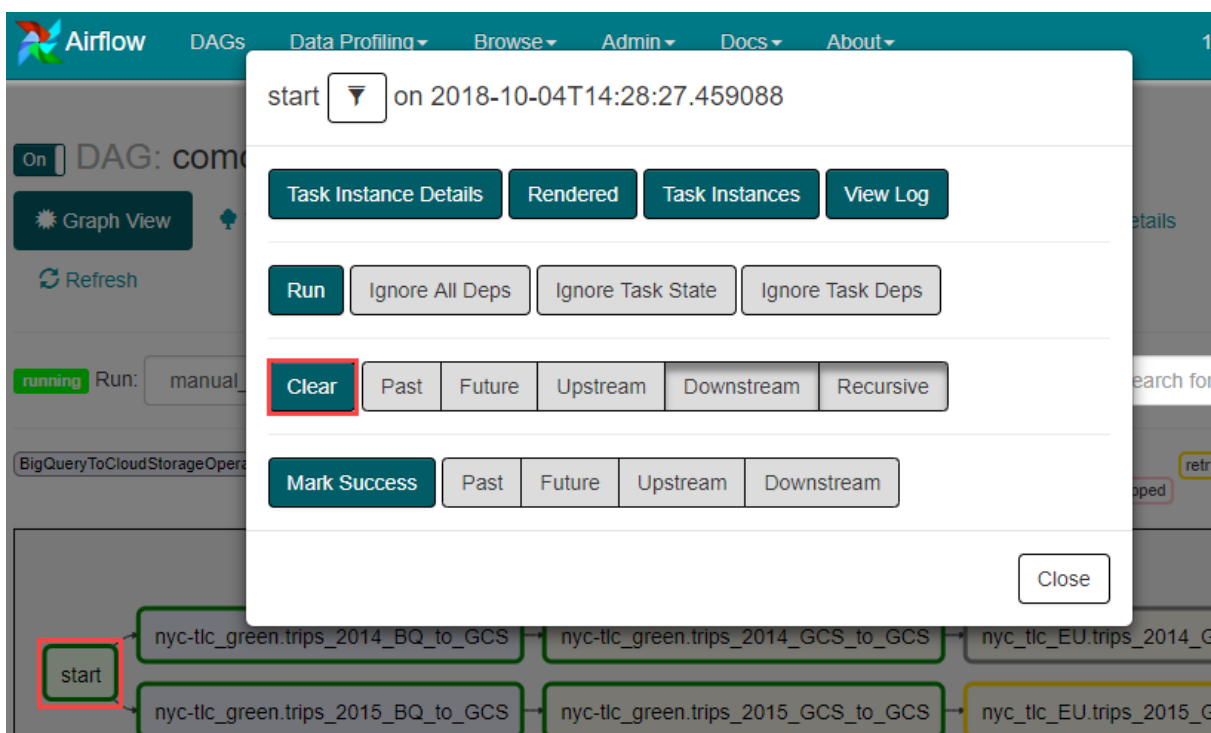


Now, in the toolbar, click **Graph View**, then mouseover the graphic for each task to see its status. Note that the border around each task also indicates the status (green border = running; red = failed, etc.).



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.



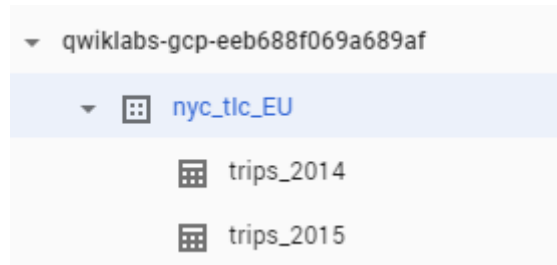
Refresh your browser while the process is running to see the most recent information.

## 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 Cloud Composer Environment

1. Return to the **Environments** page in **Composer**.
2. Select the checkbox next to your **Composer** environment.
3. Click **DELETE**.
4. Confirm the pop-up by clicking **DELETE** again.

