

Cloud Composer: Copying BigQuery Tables Across Different Locations | Google Cloud Skills Boost

Qwiklabs : 19-24 minutes

GSP283



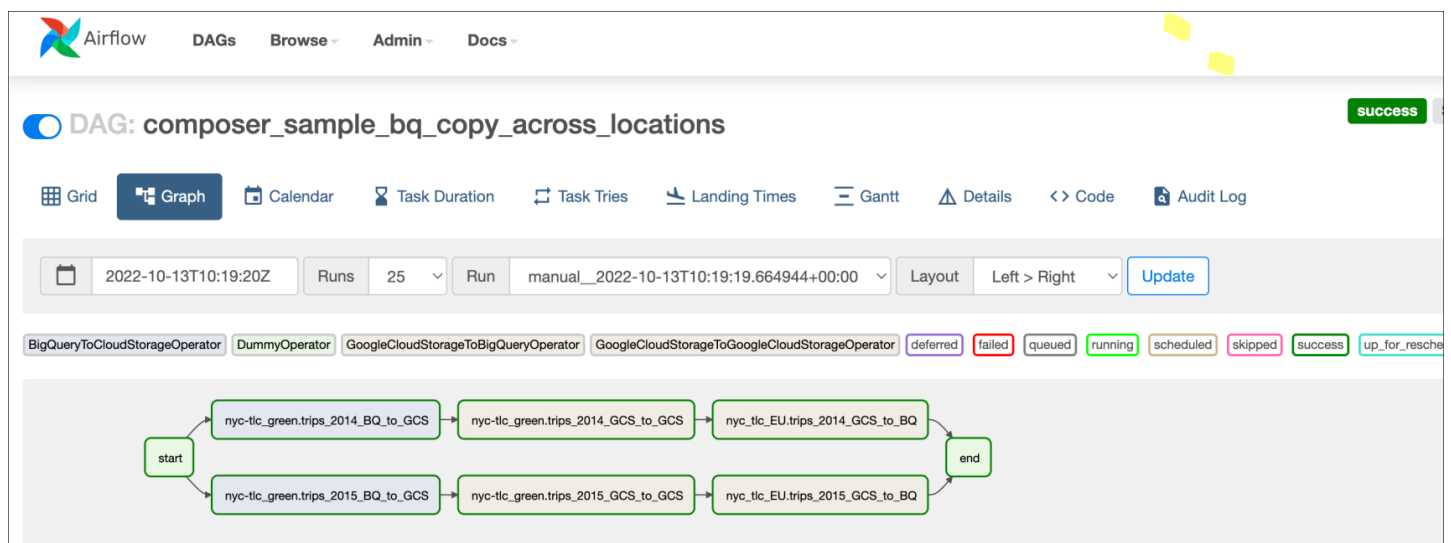
Overview

Imagine you have datasets that live in different locations around the globe. Let's say your data is in Google Cloud Storage buckets, or in BigQuery tables. How can you organize that data so that it gets consolidated and analyzed, and give you insights on your business?

Cloud Composer can help you build workflows and move your data between regions and storage systems, with an intuitive graphical view. Among others, it has templates for easy and reliable data moving between BigQuery and Cloud Storage, both ways.

In this advanced lab, you'll be able to explore that. 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



Setup

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:

- The **Open Google Console** button
- Time remaining
- The temporary credentials that you must use for this lab
- Other information, if needed, to step through this lab

2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

Tip: Arrange the tabs in separate windows, side-by-side.

Note: If you see the **Choose an account** dialog, click **Use Another Account**.

3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.

4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

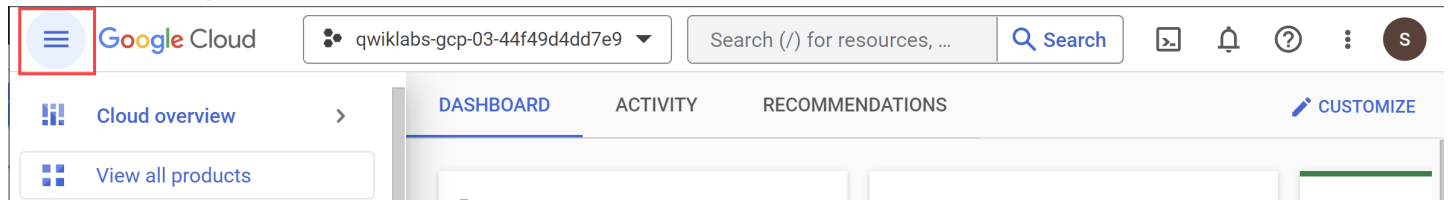
Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.

5. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.


After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell**  at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **PROJECT_ID**. The output contains a line that declares the **PROJECT_ID** for this session:

Your Cloud Platform project in this session is set to YOUR_PROJECT_ID

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3. Click **Authorize**.

4. Your output should now look like this:

Output:

```
ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net To set the active account, run: $ gcloud
config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Output:

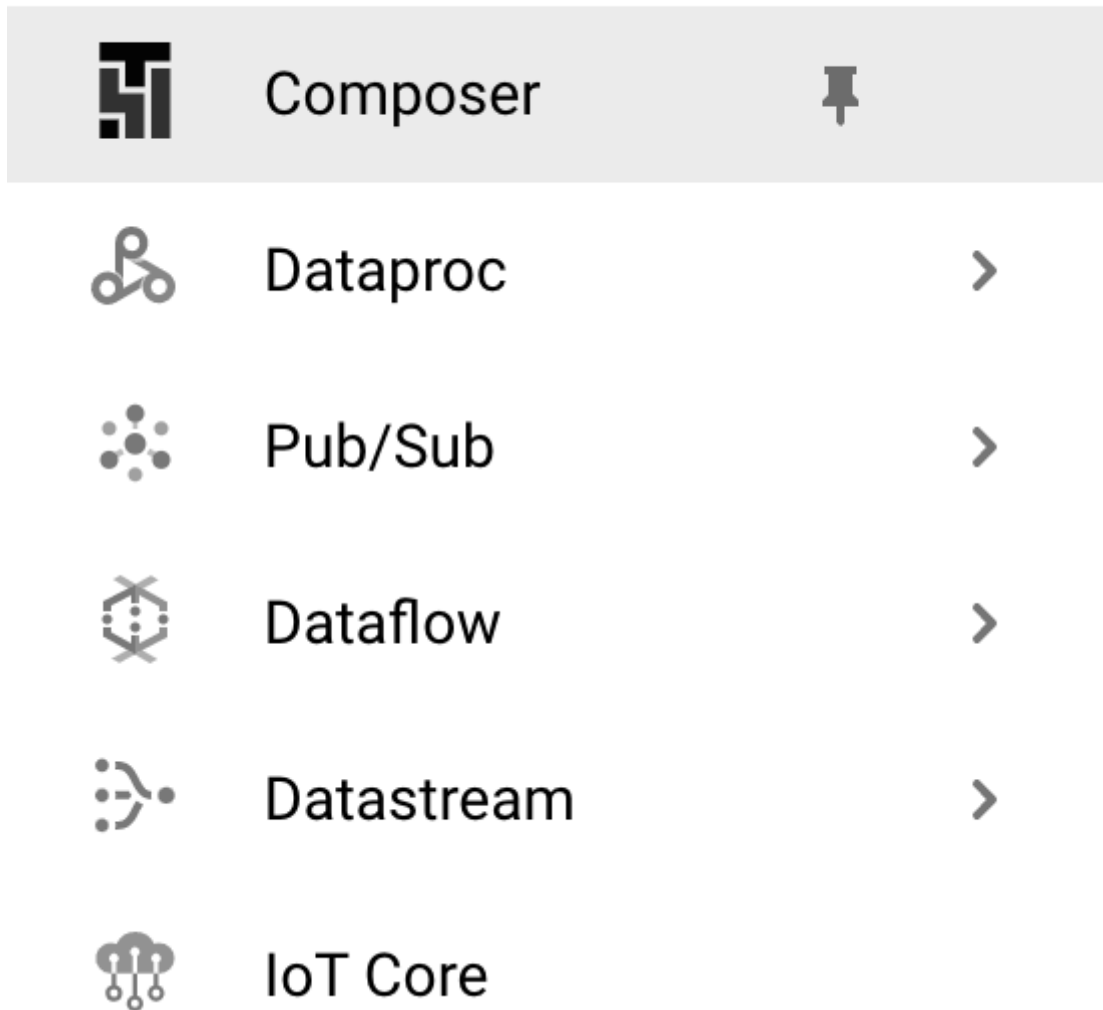
```
[core] project = <project_ID>
```

Example output:

```
[core] project = qwiklabs-gcp-44776a13dea667a6 Note: For full documentation of gcloud, in Google Cloud,
refer to the gcloud CLI overview guide.
```

Task 1. Create Cloud Composer environment

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

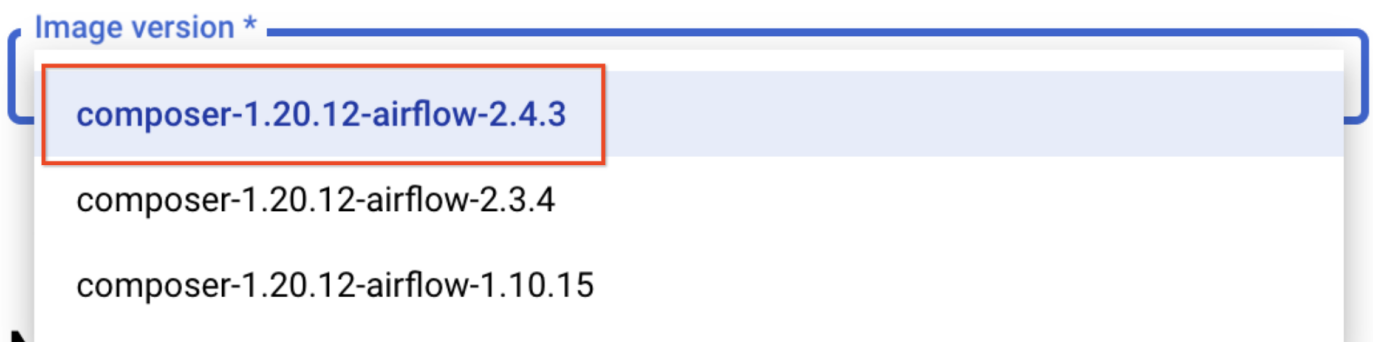


2. Then click **Create environment**.

3. In dropdown menu, select **Composer 1**.

4. Set the following parameters for your environment:

- **Name:** composer-advanced-lab
- **Location:** us-east1
- **Image Version:** composer-1.20.12-airflow-2.4.3



Node configuration

- **Zone:** us-east1-c

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.

Task 2. Create Cloud Storage buckets

1. Create two Cloud Storage Multi-Regional buckets.
2. 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)

Click **Confirm** for Public access will be prevented pop-up if prompted.

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

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

Create two Cloud Storage buckets.

Task 3. 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**.

The screenshot shows the Google Cloud Explorer interface. The top bar includes a search icon, the word 'Explorer', a '+ ADD' button, and a back arrow. Below this is a search bar with the placeholder 'Type to search'. The main area displays 'Viewing workspace resources.' and a link 'SHOW STARRED ONLY'. A resource named 'qwiklabs-gcp-02-ac54ebd89d8d' is listed with a star icon and a three-dot menu icon. A tooltip menu is open over the three-dot icon, showing 'Create dataset' and 'Refresh contents' options. The right side of the interface shows a 'Welcome to your SQL' message and a 'Get started' section with a 'View actions' button. At the bottom, there is a button that says '+ COMPOSE A NEW QUERY'.

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-02-742d72230428

[CHANGE](#)

Dataset ID *

nyc_tlc_EU

Letters, numbers, and underscores allowed

Location type ?

☐ Region

Lower latency within a single region

☒ Multi-region

Highest availability across largest area

Multi-region *

EU (multiple regions in European Union) ▼

Default table expiration

☐ Enable table expiration ?

Default maximum table age

Days

Advanced options ▼

CREATE DATASET

CANCEL

6. Click **CREATE DATASET**.

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

Create a dataset.

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. 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_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. 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.

The screenshot shows the 'Environment details' page for a Composer environment named 'composer-advanced-lab'. The environment is running. The page has tabs for MONITORING, ENVIRONMENT CONFIGURATION (selected), AIRFLOW CONFIGURATION OVERRIDES, ENVIRONMENT VARIABLES, LABELS, and PYPI PACKAGES. Below the tabs are buttons for EDIT and UPGRADE IMAGE VERSION. The main content area displays various configuration details in a table-like format:

Name	composer-advanced-lab
Zone	us-central1-a
Service account	133321383364-compute@developer.serviceaccount.com
Google API scopes	https://www.googleapis.com/auth/cloud-platform
GKE cluster	projects/qwiklabs-gcp-02-7000ba500708/zones/us-central1-a/clusters/us-central1-composer-advanc-6f44c1aa-gke
Details	view cluster details
Workloads	view cluster workloads
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) EDIT
Network configuration	
VPC-native	Disabled
Network ID	projects/qwiklabs-gcp-02-7000ba500708/global/networks/default
Subnetwork ID	-
Private environment	Disabled
Web server configuration	
Network access control	All IP addresses have access (default) EDIT
Machine type	composer-n1-webserver-2 (2 vCPU, 1.6 GB memory) EDIT
DAGs folder	gs://us-central1-composer-advanc-6f44c1aa-bucket/dags
Airflow web UI	https://v4dv945cd82a31f5p-tp.appspot.com
Stackdriver	view logs
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.

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.

- 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. 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:

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-east1-composer-advanced-YOURDAGSBUCKET-bucket. DAGS_BUCKET=<your DAGs bucket name>

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

Task 8. 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
table_list_file_path	/home/airflow/gcs/dags/bq_copy_eu_to_us_sample.csv	CSV file listing source and target tables, including dataset
gcs_source_bucket	{UNIQUE ID}-us	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
```

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-east1 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-east1 variables -- \ set gcs_source_bucket {UNIQUE ID}-us
gcloud composer environments run composer-advanced-lab \ --location us-east1 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-east1 variables -- \ get
gcs_source_bucket Note: Make sure to set all three Airflow variables used by the DAG.
```

Task 9. 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.

Task 10. 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><div></div><div></div></div>	<div><div></div><div>Name</div><div></div></div>	<div><div></div><div>Location</div></div>	<div><div></div><div>Creation time</div></div>	<div><div></div><div>Update time</div></div>	<div><div></div><div>Airflow webserver</div></div>	<div><div></div><div>Logs</div></div>	<div><div></div><div>DAGs folder</div></div>	<div><div></div><div>Labels</div></div>
<div><div></div><div></div></div>	<div><div></div><div>composer-advanced-lab</div></div>	<div><div></div><div>us-central1</div></div>	<div><div></div><div>5/12/21, 7:38 PM</div></div>	<div><div></div><div>5/12/21, 7:54 PM</div></div>	<div><div></div><div><div><div></div><div>Airflow</div></div></div></div>	<div><div></div><div><div><div></div><div>Logs</div></div></div></div>	<div><div></div><div><div><div></div><div>DAGs</div></div></div></div>	<div><div></div><div>None</div></div>

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.

Airflow

DAGs

Browse

Admin

Docs

11:17 UTC

S-

Choose File

No file chosen

Import Variables

List Variable

Search

+

Actions

Record Count: 3

<input type="checkbox"/>	Key	Val	Description	Is Encrypted
<input type="checkbox"/>	<div><div>gcs_dest_bucket</div></div>	qwiklabs-gcp-00-95008...		True
<input type="checkbox"/>	<div><div>gcs_source_bucket</div></div>	qwiklabs-gcp-00-95008...		True
<input type="checkbox"/>	<div><div>table_list_file_path</div></div>	/home/airflow/gcs/dags...		True

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 :

DAGs

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

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

Uploading the DAG and dependencies to Cloud Storage

Exploring DAG runs

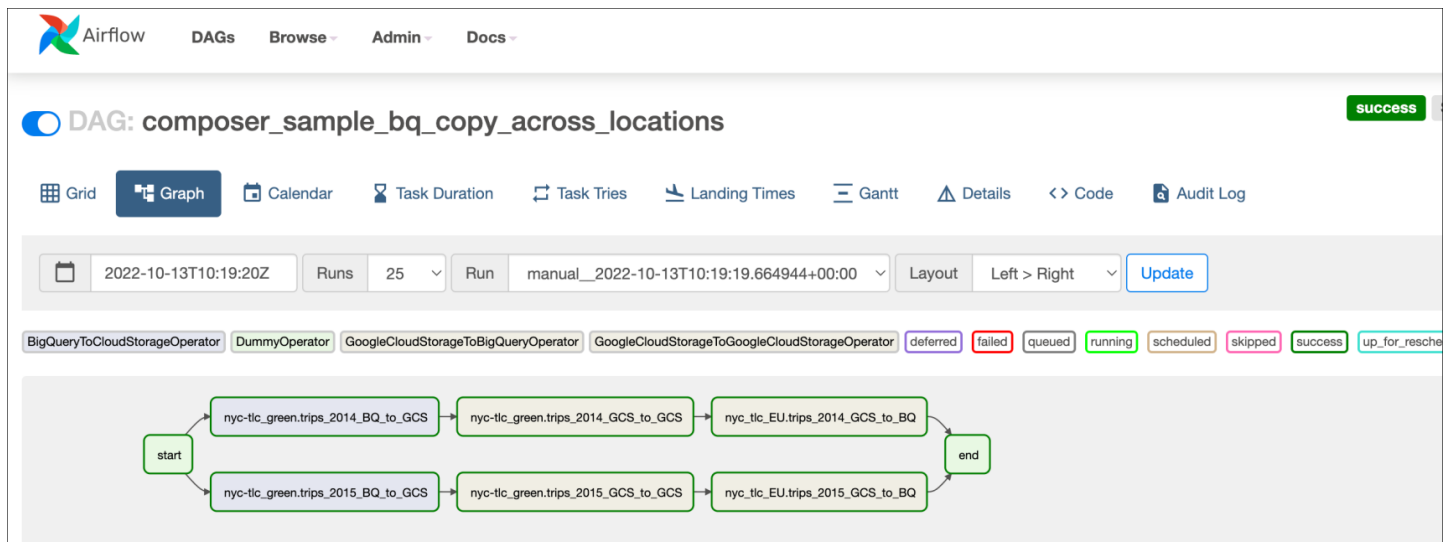
When you upload your DAG file to the DAGs folder in Cloud Storage, Cloud Composer parses the file. If no errors are found, the name of the workflow appears in the DAG listing, and the workflow is queued to run immediately if the schedule conditions are met, in this case, None as per the settings.

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

DAGs

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

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 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.

Congratulations!

You've have completed this advanced lab and copied 2 tables programmatically from US to EU! This lab is based on this [blog post](#) by David Sabater Dinter.

Finish your quest

This self-paced lab is part of the **Data Engineering** quest. A quest is a series of related labs that form a learning path. Completing this quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. [Enroll in this quest](#) or any quest that contains this lab and get immediate completion credit. See the [Google Cloud Skills Boost catalog](#) to see all available quests.

Take your next lab

Continue your Quest with [Predict Visitor Purchases with a Classification Model in BQML](#), or check out these suggestions:

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 the Apache dev and commits mailing lists (send emails to dev-subscribe@airflow.incubator.apache.org and commits-subscribe@airflow.incubator.apache.org to subscribe to each)
- 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](#).

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated June 07, 2023

Lab Last Tested June 07, 2023

Copyright 2023 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.