

Cloud Composer: Copying BigQuery Tables Across Different Locations

GSP283

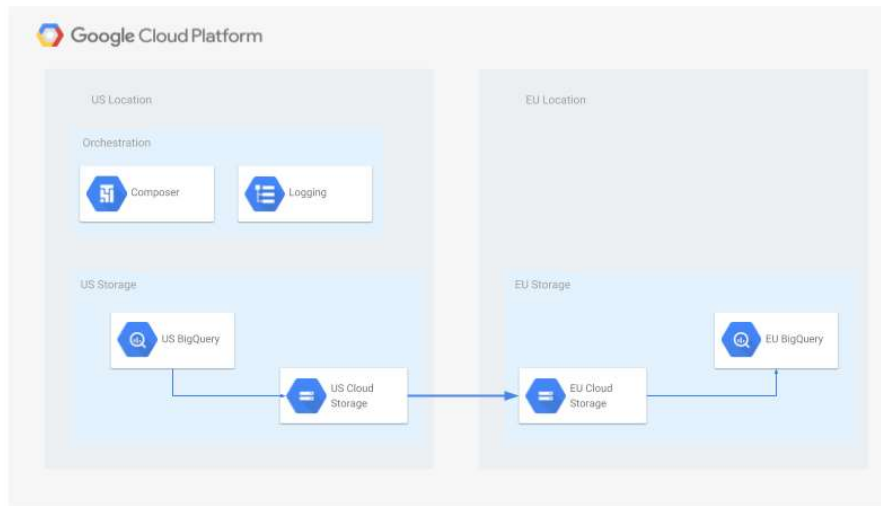


Overview

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



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 Qwiklabs 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.

What you need

To complete this lab, you need:

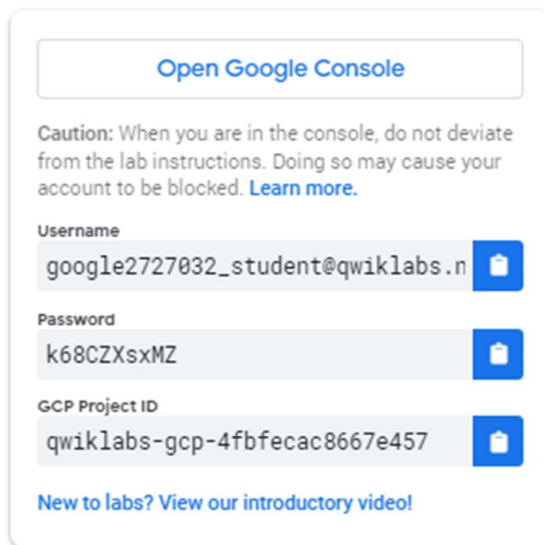
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

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

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

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

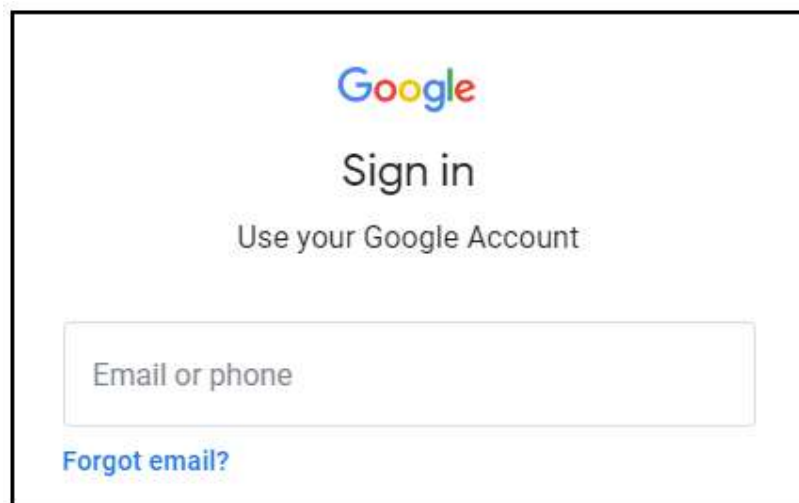
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



This panel contains the following information:

- Open Google Console** (button)
- Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)
- Username:** google2727032_student@qwiklabs.n (with a copy icon)
- Password:** k68CZXsxMZ (with a copy icon)
- GCP Project ID:** qwiklabs-gcp-4fbfecac8667e457 (with a copy icon)
- [New to labs? View our introductory video!](#)

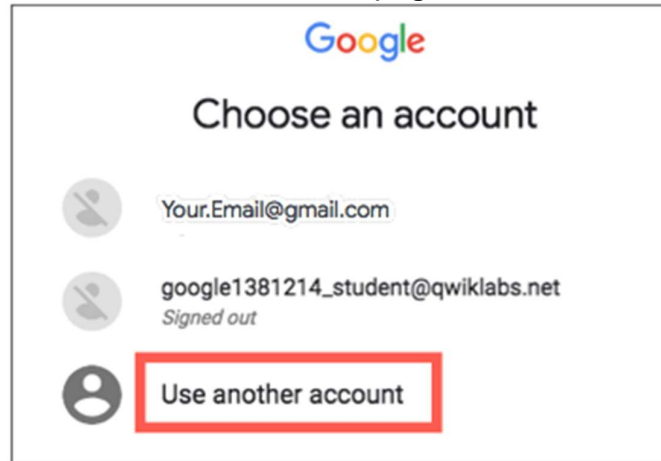
2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



The sign-in page displays the Google logo, the text "Sign in", and "Use your Google Account". It features a text input field labeled "Email or phone" and a link for "Forgot email?" below it.

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

If you see the **Choose an account** page, click **Use Another**



Account.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

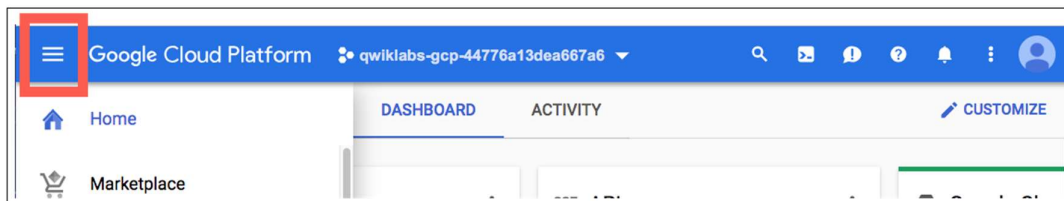
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. 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.

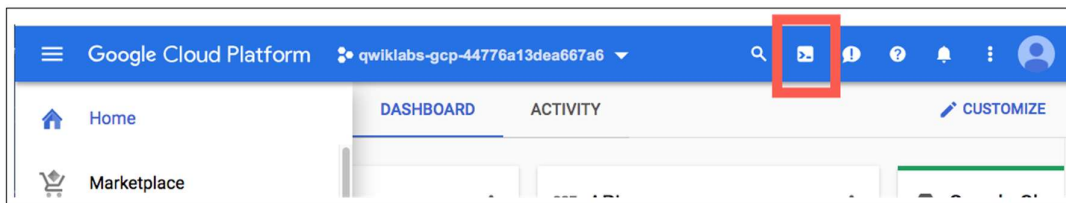
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



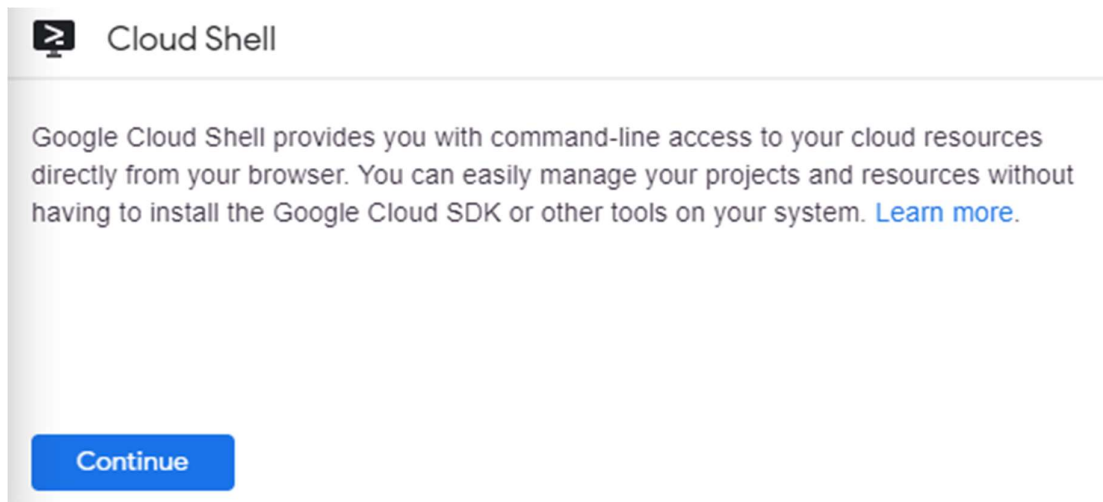
Activate Cloud Shell

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.

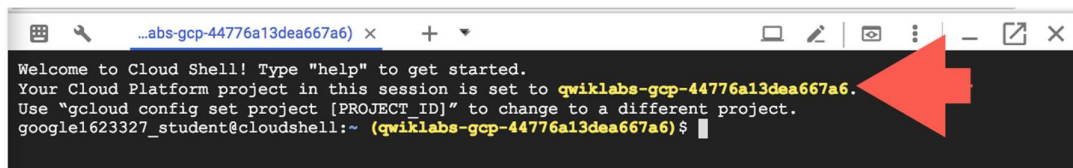
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



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

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
- <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
- google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

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


(Example output)

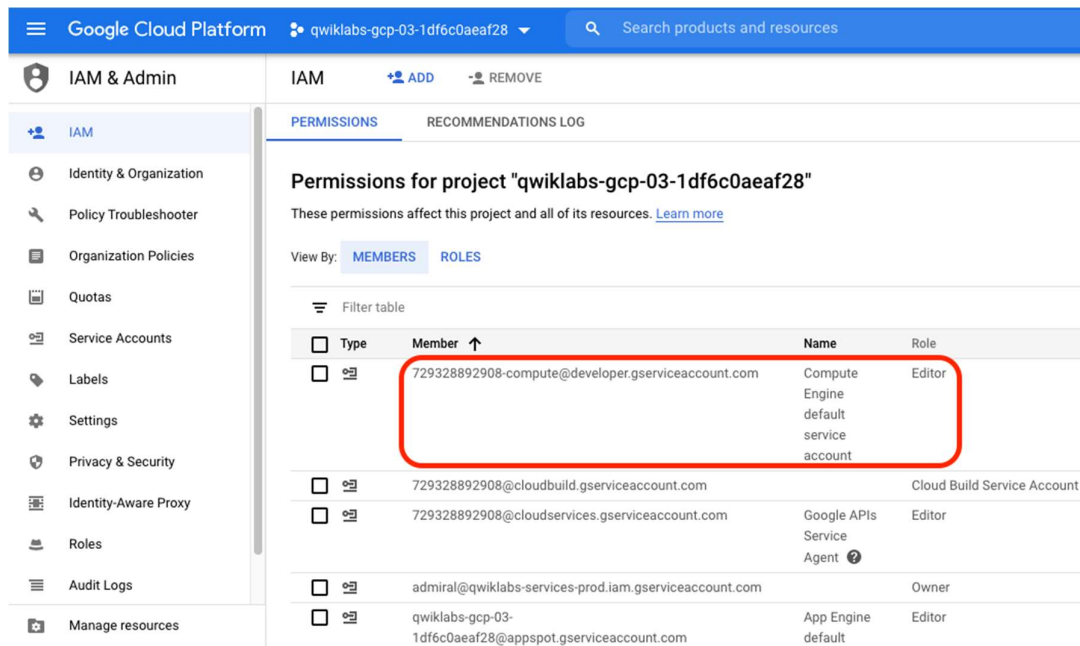
```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the [gcloud command-line tool overview](#).

Check project permissions

Before you begin your work on Google Cloud, you need to ensure that your project has the correct permissions within Identity and Access Management (IAM).

1. In the Google Cloud console, on the **Navigation menu** () , click **IAM & Admin > IAM**.
2. Confirm that the default compute Service Account `{project-number}-compute@developer.gserviceaccount.com` is present and has the `editor` role assigned. The account prefix is the project number, which you can find on **Navigation menu > Home**.



Google Cloud Platform | qwiklabs-gcp-03-1df6c0aeaf28 | Search products and resources

IAM & Admin

IAM [ADD](#) [REMOVE](#)

PERMISSIONS **RECOMMENDATIONS LOG**

Permissions for project "qwiklabs-gcp-03-1df6c0aeaf28"

These permissions affect this project and all of its resources. [Learn more](#)

View By: **MEMBERS** **ROLES**

Filter table

Type	Member ↑	Name	Role
<input type="checkbox"/>	729328892908-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor
<input type="checkbox"/>	729328892908@cloudbuild.gserviceaccount.com		Cloud Build Service Account
<input type="checkbox"/>	729328892908@cloudservices.gserviceaccount.com	Google APIs Service Agent	Editor
<input type="checkbox"/>	admiral@qwiklabs-services-prod.iam.gserviceaccount.com		Owner
<input type="checkbox"/>	qwiklabs-gcp-03-1df6c0aeaf28@appspot.gserviceaccount.com	App Engine default	Editor

If the account is not present in IAM or does not have the `editor` role, follow the steps below to assign the required role.

- In the Google Cloud console, on the **Navigation menu**, click **Home**.

- Copy the project number (e.g. 729328892908).
- On the **Navigation menu**, click **IAM & Admin > IAM**.
- At the top of the **IAM** page, click **Add**.
- For **New members**, type:

```
{project-number}-compute@developer.gserviceaccount.com
```

Replace {project-number} with your project number.

- For **Role**, select **Project (or Basic) > Editor**. Click **Save**.

The screenshot shows the Google Cloud Platform IAM & Admin console. The left sidebar contains the navigation menu with options like Identity & Organization, Policy Troubleshooter, Organization Policies, Quotas, Service Accounts, Labels, Settings, Privacy & Security, Identity-Aware Proxy, Roles, Audit Logs, and Manage resources. The main content area is titled 'IAM' and shows the 'PERMISSIONS' tab for the project 'qwiklabs-gcp-03-1df6c0aeaf28'. A modal dialog is open on the right, titled 'Add members to "qwiklabs-gcp-03-1df6c0aeaf28"'. The dialog contains the following elements:

- Add members, roles to "qwiklabs-gcp-03-1df6c0aeaf28" project**: A heading for the modal.
- Enter one or more members below. Then select a role for these members to grant them access to your resources. Multiple roles allowed. [Learn more](#)**: A sub-heading with a link.
- New members**: A text input field containing the email address '729328892908-compute@developer.gserviceaccount.com'.
- Role**: A dropdown menu set to 'Editor'.
- Condition**: A link to 'Add condition'.
- + ADD ANOTHER ROLE**: A button to add more roles.
- ☐ **Send notification email**: A checkbox with a description: 'This email will inform members that you've granted them access to this role for "qwiklabs-gcp-03-1df6c0aeaf28"'
- SAVE** and **CANCEL** buttons at the bottom.

Create Cloud Composer environment

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

BIG DATA



Composer



Dataproc



Pub/Sub



Dataflow



IoT Core



BigQuery



Data Catalog

Then click **Create environment**.

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 dataset.

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

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.

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

BigQuery destination dataset

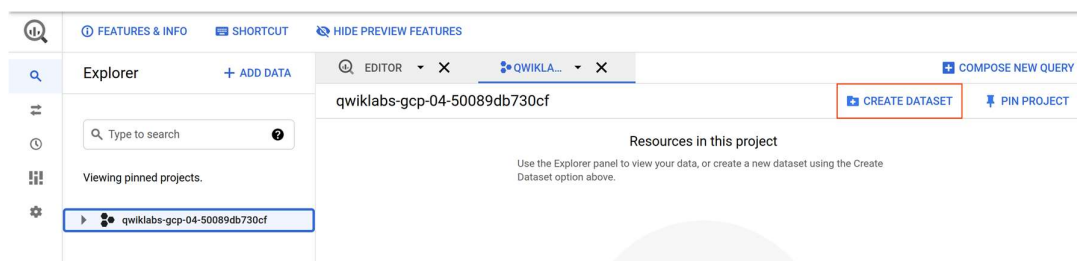
Create the destination BigQuery Dataset in EU from the BigQuery new web UI.

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 on your Qwiklabs project ID:



Click **Create Dataset**. Use the name **nyc_tlc_EU** and Data location **EU**.

Create dataset

Dataset ID

Data location (Optional) ?

Default table expiration ?

- ☒ Never
- ☐ Number of days after table creation:

Encryption

Data is encrypted automatically. Select an encryption key management solution.

- ☒ Google-managed key
No configuration required
- ☐ Customer-managed key
Manage via Google Cloud Key Management Service

Create dataset

Cancel

Click **Create dataset**.

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

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 command.

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

[Task Instance](#) - A specific run of a task; characterised 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.

The screenshot shows the 'Environment details' page for a Google Cloud Composer environment named 'composer-advanced-lab'. The page indicates the environment is running. It features several tabs: MONITORING (BETA), ENVIRONMENT CONFIGURATION (selected), AIRFLOW CONFIGURATION OVERRIDES, ENVIRONMENT VARIABLES, LABELS, and PYTHON PACKAGES. Below the tabs are buttons for 'EDIT' and 'UPGRADE IMAGE VERSION (BETA)'. The main content area displays a list of configuration details for the environment, including its name, zone, service account, Google API scopes, GKE cluster information, image version, Python version, network tags, worker nodes configuration, Cloud SQL configuration, network configuration, private environment status, web server configuration, DAGs folder, Airflow web UI URL, Stackdriver logs, and creation/update timestamps.

Name	composer-advanced-lab
Zone	us-central1-a
Service account	15332138384-compute@developer.gserviceaccount.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-advanced-lab-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-advanced-lab-bucket/dags
Airflow web UI	https://4d9f45cd052a31f5p1p.apigee.net
Stackdriver	view logs
Created	Wed Sep 09 2020 22:35:47 GMT+0530 (India Standard Time)
Updated	Wed Sep 09 2020 22:52:45 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 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
<code>gcs_dest_bucket</code>	<code>{UNIQUE ID}-eu</code>	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-centrall 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-centrall 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 new window icon.

Google Cloud Platform qwiklabs-gcp-1d77f959f80eb959

Composer Environments [+ CREATE](#) [DELETE](#)

Filter environments

<input type="checkbox"/>	Name ↑	Location	Creation time	Update time	Airflow webserver	DAGs folder	Labels
<input checked="" type="checkbox"/>	composer-advanced-lab	us-central1	9/4/18, 3:19 PM	9/4/18, 3:41 PM	↗	■	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

No file chosen

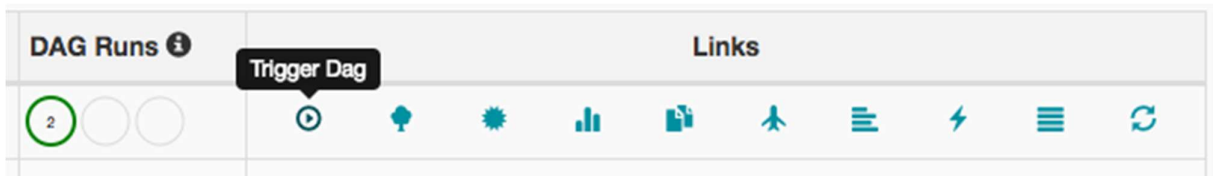
List (3) [Create](#) [Add Filter](#) [With selected](#)

<input type="checkbox"/>		Key	Val
<input type="checkbox"/>	 	gcs_dest_bucket	6552634-eu
<input type="checkbox"/>	 	gcs_source_bucket	6552634-us
<input type="checkbox"/>	 	master_file_path	/home/airflow/gcs/dags/bq_copy_eu_to_us_master.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`:



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

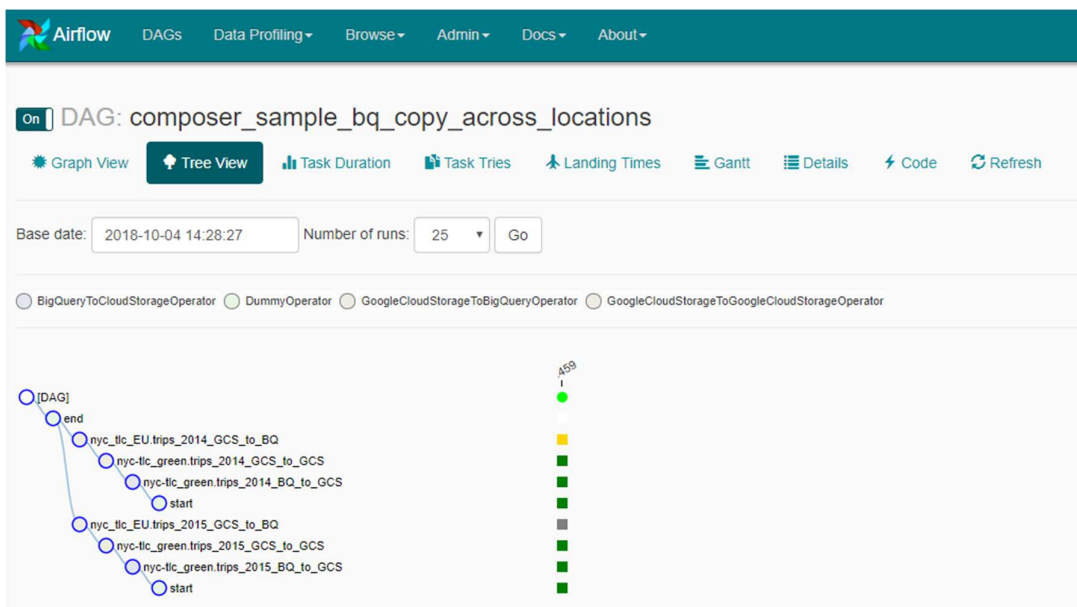
DAGs

Search:

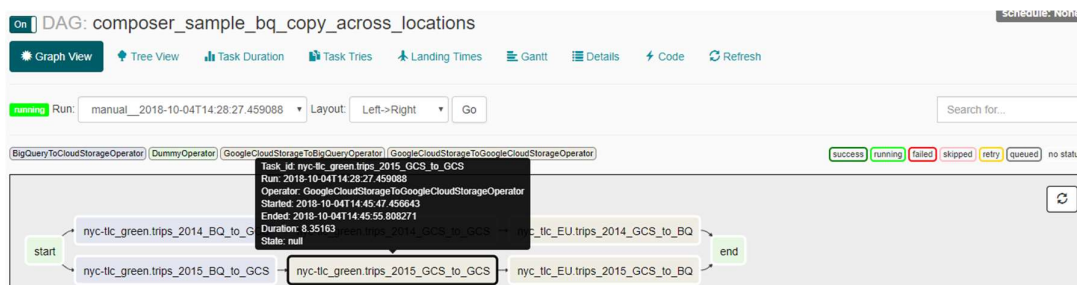
	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On composer_sample_bq_copy_across_locations	None	airflow				

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.

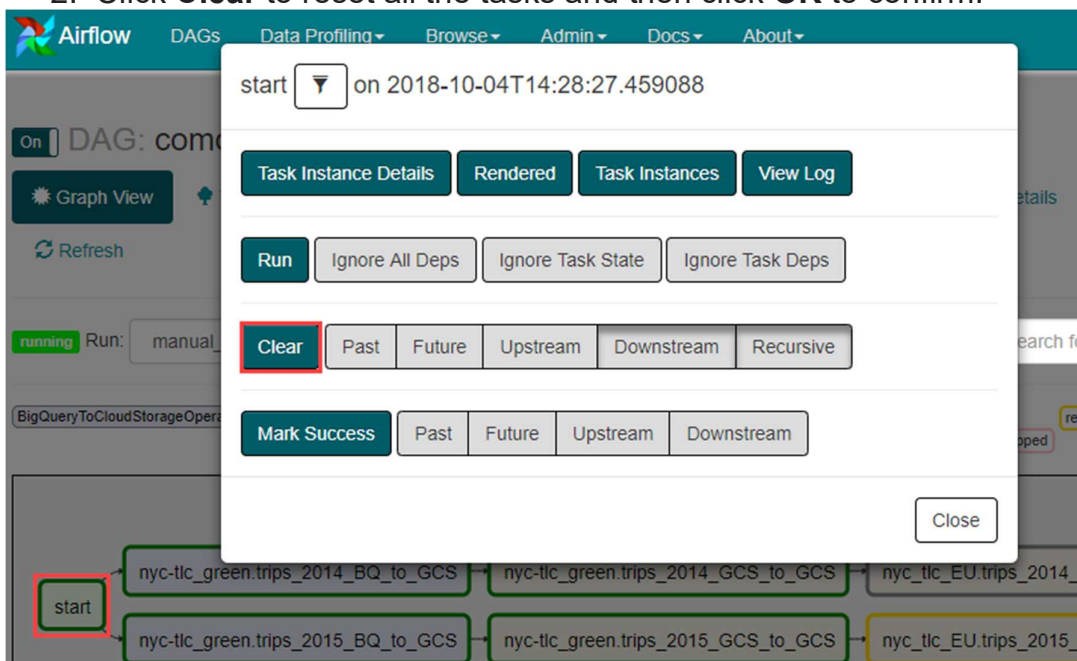


Now, in the toolbar, click **Graph View**, then mouseover the graphic for each task to see its status. Note that the border around each task also indicates the status (green border = running; red = failed, etc.).



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.



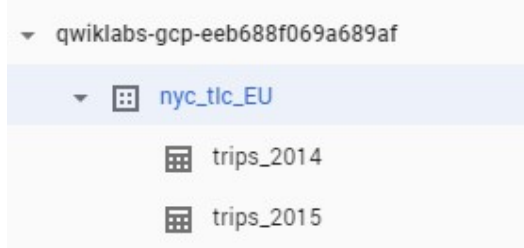
Refresh your browser while the process is running to see the most recent information.

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 **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.



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.

Take Your Next Lab

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

- [Predict Housing Prices with Tensorflow and AI Platform](#)

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 & 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 March 23, 2021

Lab Last Tested March 23, 2021

Copyright 2021 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.