Why use Cloud Composer?

Cloud Composer is a fully managed workflow orchestration service, enabling you to create workflows that span across clouds and on-premises data centers. Built on the popular Apache Airflow open source project and operated using the Python programming language, Cloud Composer is free from lock-in and easy to use. By using Cloud Composer instead of a local instance of Apache Airflow, users can benefit from the best of Airflow with no installation or management overhead.

Workflows, DAGs, and tasks

In data analytics, a **workflow** represents a series of tasks for ingesting, transforming, analyzing, or utilizing data. In Airflow, workflows are created using DAGs, or "Directed Acyclic Graphs".

A **DAG** is a collection of tasks that you want to schedule and run, organized in a way that reflects their relationships and dependencies. DAGs are created in Python scripts, which define the DAG structure (tasks and their dependencies) using code.

Each **task** in a DAG can represent almost anything—for example, one task might perform any of the following functions:

- Preparing data for ingestion
- Monitoring an API
- Sending an email
- Running a pipeline

A DAG shouldn't be concerned with the function of each constituent task—its purpose is to ensure that each task is executed at the right time, in the right order, or with the right issue handling.

Environments

To run workflows, you first need to create an **environment**. Airflow depends on many micro-services to run, so Cloud Composer provides Google Cloud components to run your workflows. These components are collectively known as a **Cloud Composer environment**.

Environments are self-contained Airflow deployments based on Google Kubernetes Engine, and they work with other Google Cloud services using connectors built into Airflow. You can create one or more environments in a single Google Cloud project. You can create Cloud Composer environments in any supported region.

Can I use my own database as the Airflow Metadata DB?

Cloud Composer uses a managed database service for the Airflow Metadata DB. It is not possible to use a user-provided database as the Airflow Metadata DB.

Can I use my own cluster as a Cloud Composer cluster?

Cloud Composer uses Google Kubernetes Engine service to create, manage and delete environment clusters where Airflow components run. These clusters are fully managed by Cloud Composer.

Are Cloud Composer environments zonal or regional?

Cloud Composer 1 environments are zonal.

Cloud Composer 2 environments have a zonal Airflow Metadata DB and a regional Airflow scheduling & execution layer. Airflow schedulers, workers and web servers run in the Airflow execution layer.

What version of Apache Airflow does Cloud Composer use?

Cloud Composer supports both Airflow 1 and Airflow 2.

Cloud Composer environments are based on Cloud Composer images. When you create an environment, you can select an image with a specific Airflow version.

Cloud Composer: Copying BigQuery Tables Across Different Locations

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

Create Cloud Composer 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.

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**.

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 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*, ...

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.

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.