An Introduction to Cloud Composer 2.5

Overview

Workflows are a common theme in data analytics - they involve ingesting, transforming, and analyzing data to figure out the meaningful information within. In Google Cloud, the tool for hosting workflows is Cloud Composer which is a hosted version of the popular open source workflow tool Apache Airflow.

In this lab, you use the Google Cloud console to set up a Cloud Composer environment. You then use Cloud Composer to go through a simple workflow that verifies the existence of a data file, creates a Cloud Dataproc cluster, runs an Apache Hadoop wordcount job on the Cloud Dataproc cluster, and deletes the Cloud Dataproc cluster afterwards.

What you'll do

- Use the Google Cloud console to create the Cloud Composer environment
- View and run the DAG (Directed Acyclic Graph) in the Airflow web interface
- View the results of the wordcount job in storage

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** (☰), select **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 > Cloud Overview > Dashboard**.

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

1. In the Google Cloud console, on the **Navigation menu**, click **Cloud Overview > Dashboard**.

2. Copy the project number (e.g. `729328892908`).

3. On the **Navigation menu**, select **IAM & Admin** > **IAM**.

4. At the top of the roles table, below **View by Principals**, click **Grant Access**.

5. For **New principals**, type:

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

6. Replace `{project-number}` with your project number.

7. For **Role**, select **Project** (or Basic) > **Editor**.

8. Click **Save**.

Task 1. Environment Tasks

1. On the Google Cloud Console title bar, click **Activate Cloud Shell**. If prompted, click **Continue**.

2. Run the following commands to assign the **Composer Worker** role to the Compute Developer Service Acocunt.

```
export PROJECT_ID=$(gcloud config get-value project)
```

```
export PROJECT_NUMBER=$(gcloud projects describe
$PROJECT_ID --format="value(projectNumber)")
```

```
gcloud projects add-iam-policy-binding qwiklabs-gcp-01-
eb7a1a49de7a \
--member=serviceAccount:$PROJECT_NUMBER-
compute@developer.gserviceaccount.com \
--role=roles/composer.worker
```

3. Run the following commands to ensure that the required APIs are enabled cleanly in your project.

```
gcloud services disable composer.googleapis.com
gcloud services disable artifactregistry.googleapis.com
gcloud services disable container.googleapis.com
```

```
gcloud services enable artifactregistry.googleapis.com
gcloud services enable container.googleapis.com
gcloud services enable composer.googleapis.com
```

Task 2. Create Cloud Composer environment

In this section, you create a Cloud Composer environment.

Note: Before proceeding further, make sure that you have performed earlier tasks to ensure that the required APIs are successfully enabled. If not, then please perform those tasks otherwise Cloud Composer environment creation will fail.

1. On the Google Cloud console title bar, type **Composer** in the Search field, then click **Composer** in the Products & Page section.

2. Click **Create Environment** and select **Composer 3**. Set the following for your environment:

Property	**Value**
Name	highcpu
Location	`us-east4`
Image	composer-3-airflow-n.n.n-build.n (Note: select the highest number

3. Under **Environment resources**, Select **Small**.

Leave all other settings as default.

4. 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 console.

It can take 15-30 minutes for the environment to complete the setup process. Continue with the lab while the environment spins up.

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

Create a Cloud Storage bucket

Create a Cloud Storage bucket in your project. This bucket will be used as output for the Hadoop job from Dataproc.

1. Go to **Navigation menu** > **Cloud Storage** > **Buckets** and then click **+ Create**.

2. Give your bucket a universally unique name such as your Project ID, **`qwiklabs-gcp-01-eb7a1a49de7a`**, then click **Create**. If prompted `Public access will be prevented`, click **Confirm**.

Remember the Cloud Storage bucket name to use it as an Airflow variable later in the lab.

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

Task 3. Airflow and core concepts

While waiting for your Composer environment to get created, review some terms that are used with Airflow.

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 all the tasks you want to run, organized in a way that reflects 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*, ...

You can read more about the concepts in the Concepts documentation.

Task 4. Defining the workflow

Now let's discuss the workflow you'll be using. Cloud Composer workflows are comprised of DAGs (Directed Acyclic Graphs). DAGs are defined in standard Python files that are placed in Airflow's `DAG_FOLDER`. Airflow will execute the code in each file to dynamically build the `DAG` objects. You can have as many DAGs as you want, each describing an arbitrary number of tasks. In general, each one should correspond to a single logical workflow.

Below is the `hadoop_tutorial.py` workflow code, also referred to as the DAG:

See hadoop_tutorial.py

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

1. `DataprocClusterCreateOperator`: Creates a Cloud Dataproc cluster.

2. `DataProcHadoopOperator`: Submits a Hadoop wordcount job and writes results to a Cloud Storage bucket.

3. `DataprocClusterDeleteOperator`: Deletes the cluster to avoid incurring ongoing Compute Engine charges.

The tasks run sequentially, which you can see in this section of the file:

```
# Define DAG dependencies.
create_dataproc_cluster >> run_dataproc_hadoop >>
delete_dataproc_cluster
```

The name of the DAG is **composer_hadoop_tutorial**, and the DAG runs once each day:

```
with models.DAG(
        'composer_hadoop_tutorial',
        # Continue to run DAG once per day
        schedule_interval=datetime.timedelta(days=1),
        default_args=default_dag_args) as dag:
```

Because the `start_date` that is passed in to `default_dag_args` is set to `yesterday`, Cloud Composer schedules the workflow to start immediately after the DAG uploads.

Task 5. Viewing environment information

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

2. Once your environment has been created, click the name of the environment (highcpu) to see its details.

On the **Environment configuration** tab you'll see information such as the Airflow web UI URL, GKE cluster, and a link to the DAGs folder, which is stored in your bucket.

Note: Cloud Composer only schedules the workflows in the `/dags` folder.

Task 6. Using the Airflow UI

To access the Airflow web interface using the console:

1. Go back to the **Environments** page.

2. In the **Airflow webserver** column for the environment, click **Airflow**.

3. Click on your lab credentials.

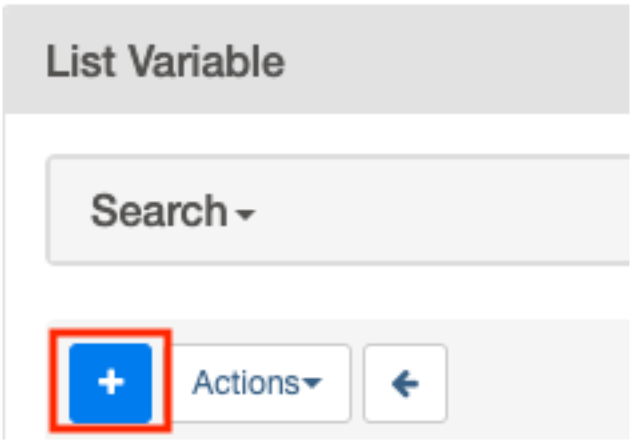
4. The Airflow web interface opens in a new browser window.

Task 7. Setting Airflow variables

Airflow variables are an Airflow-specific concept that is distinct from environment variables.

1. From the Airflow interface, select **Admin** > **Variables** from the menu bar.

2. Click **+** icon to add a new record.

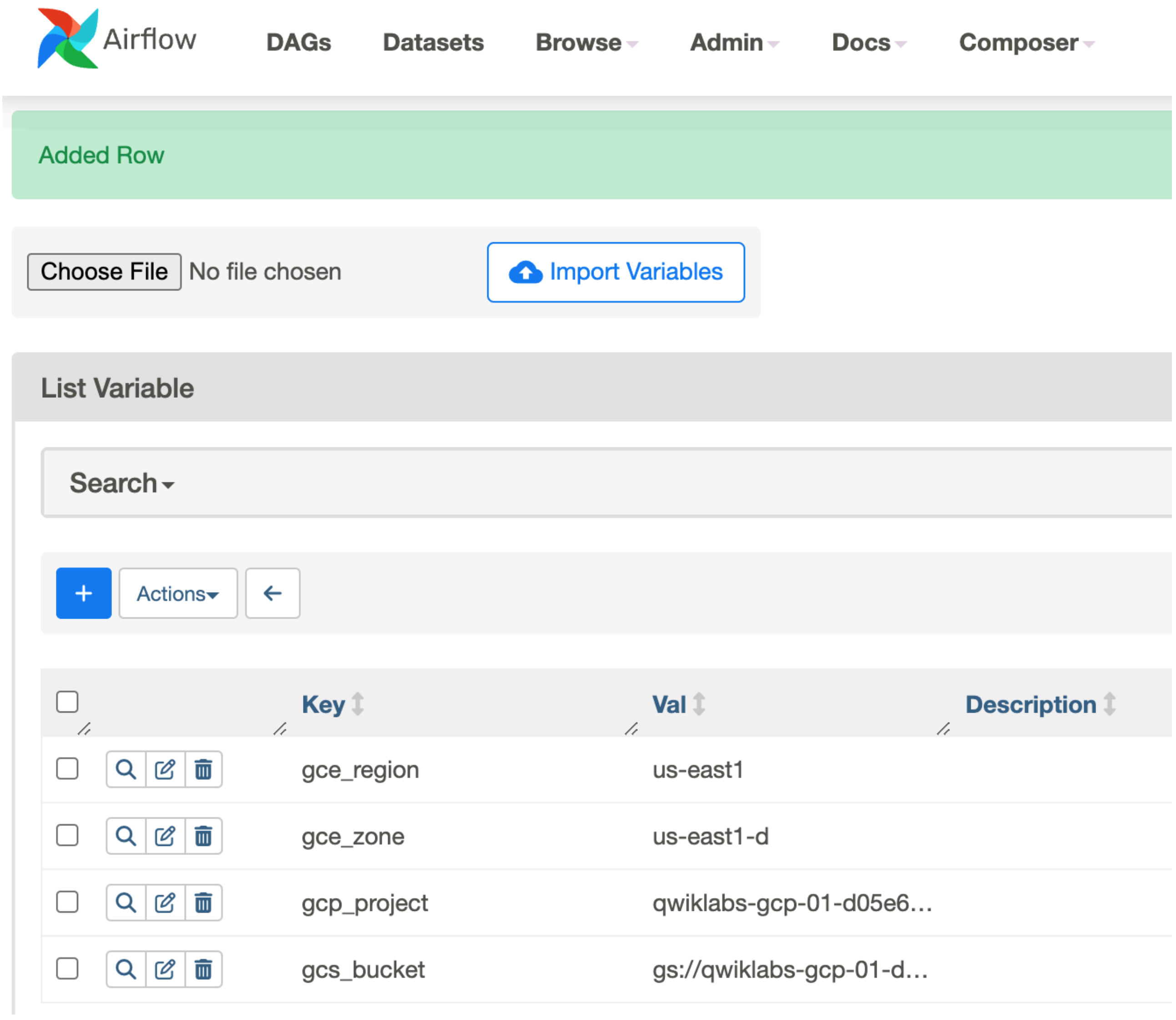


3. Create the following Airflow variables: **gcp_project**, **gcs_bucket**, **gce_zone**, and **gce_region**. Click **Save** after each variable.

Key	Val	Details
`gcp_project`	`qwiklabs-gcp-01-eb7a1a49d`	The Google Cloud Platform project you're using for this lab.
`gcs_bucket`	gs://<my-bucket>	Replace <my-bucket> with the name of the Cloud Storage bucket you made earlier. This bucket stores the output from the Hadoop jobs

`gce_zone`	`us-east4-a`	This is the Compute Engine zone where your Cloud Dataproc cluster will be created.
`gce_region`	`us-east4`	This is the Compute Engine region where your Cloud Dataproc cluster will be created.

Click **Save**. After adding first variable repeat the same process for second and third variable. Your Variables table should look like this when you're finished:



Key	Val	Description
gce_region	us-east1	
gce_zone	us-east1-d	
gcp_project	qwiklabs-gcp-01-d05e6...	
gcs_bucket	gs://qwiklabs-gcp-01-d...	

Task 8. Uploading the DAG to Cloud Storage

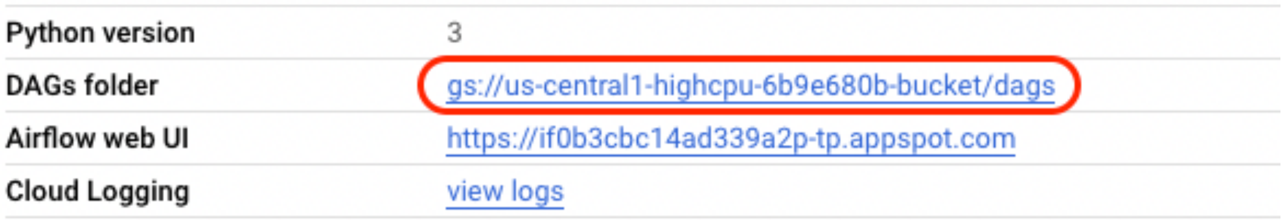
To upload the DAG:

1. In the **Cloud Shell** run the below command to upload a copy of the `hadoop_tutorial.py` file to the Cloud Storage bucket that was automatically created when you created the environment.

2. Replace `<DAGs_folder_path>` in the following command with the path to the DAGs folder:

```
gcloud storage cp gs://cloud-training/datawarehousing/
lab_assets/hadoop_tutorial.py <DAGs_folder_path>
```

- You can get the path by going to **Composer**.
- Click on the environment you created earlier and then click on the **Environment Configuration** tab to see the details of the environment.
- Find `DAGs folder` and copy the path.

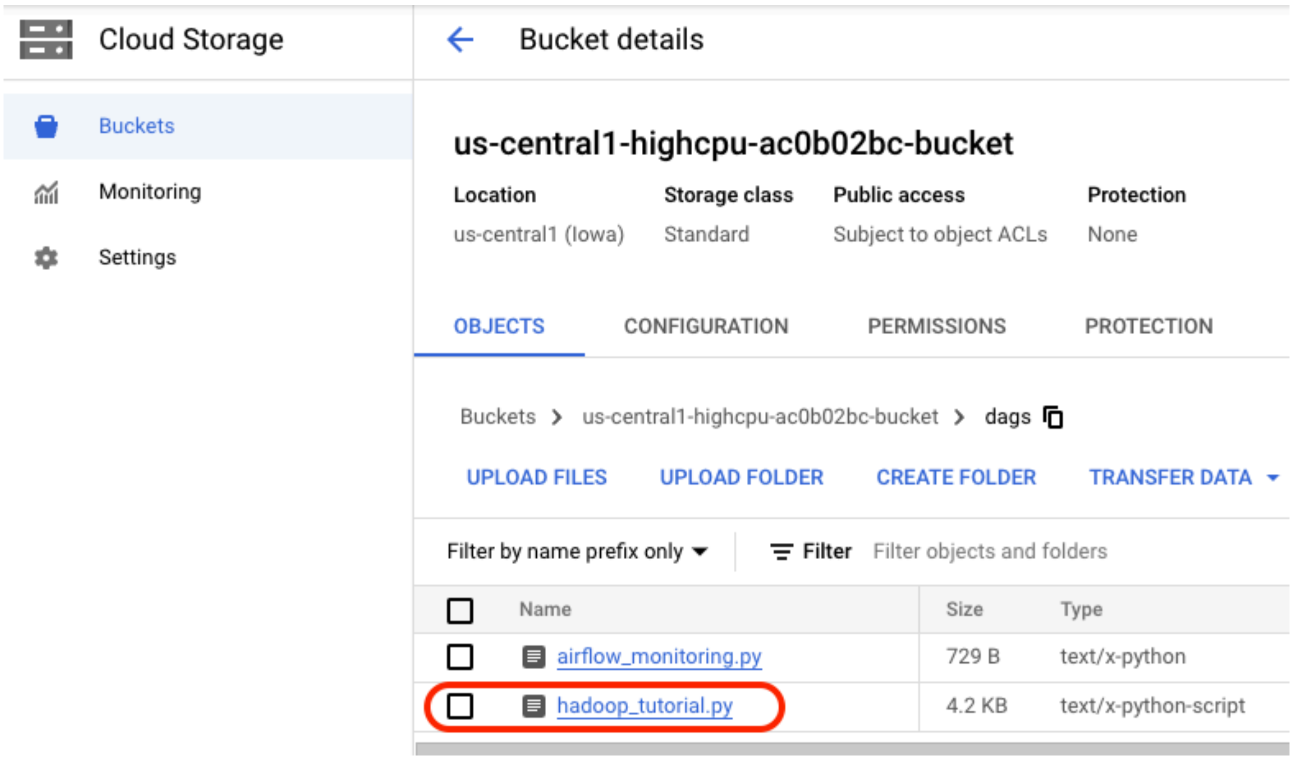


Python version	3
DAGs folder	gs://us-central1-highcpu-6b9e680b-bucket/dags
Airflow web UI	https://if0b3cbc14ad339a2p-tp.appspot.com
Cloud Logging	view logs

The revised command to upload the file will look similar to the one below:

```
gcloud storage cp gs://cloud-training/datawarehousing/
lab_assets/hadoop_tutorial.py gs://us-east4-
highcpu-0682d8c0-bucket/dags
```

3. Once the file has been successfully uploaded to the DAGs directory, open `dags` folder in the bucket and you will see the file in the **Objects** tab of the Bucket details.



us-central1-highcpu-ac0b02bc-bucket

Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Subject to object ACLs	None

Buckets > us-central1-highcpu-ac0b02bc-bucket > dags

Name	Size	Type
airflow_monitoring.py	729 B	text/x-python
hadoop_tutorial.py	4.2 KB	text/x-python-script

When a DAG file is added to the DAGs folder, Cloud Composer adds the DAG to Airflow and schedules it automatically. DAG changes occur within 3-5 minutes.

You can see the task status of the `composer_hadoop_tutorial` DAG in the Airflow web interface.

Note: You may safely ignore any message on the interface such as "**The scheduler does not appear to be running...**". The Airflow web interface will update as the DAG progresses.

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.

1. Make sure that you're on the DAGs tab in the Airflow web interface. It takes several minutes for this process to complete. Refresh your browser to make sure you're looking at the latest information.

2. In Airflow, click **composer_hadoop_tutorial** to open the DAG details page. This page includes several representations of the workflow tasks and dependencies.

3. In the toolbar, click **Graph**. 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.).

4. Click the "Refresh" link to make sure you're looking at the most recent information. The borders of the processes change color as the state of the process changes

Note: If your Dataproc cluster already exists, you can run the workflow again to reach the success state by clicking `create_dataproc_cluster` graphic and then click **Clear** to reset the three tasks and click **OK** to confirm.

5. Once the status for **create_dataproc_cluster** has changed to "running", go to **Navigation menu > Dataproc**, then click on:

    - **Clusters** to monitor cluster creation and deletion. The cluster created by the workflow is ephemeral: it only exists for the duration of the workflow and is deleted as part of the last workflow task.

    - **Jobs** to monitor the Apache Hadoop wordcount job. Click the Job ID to see job log output.

6. Once Dataproc gets to a state of "Running", return to Airflow and click **Refresh** to see that the cluster is complete.

When the `run_dataproc_hadoop` process is complete, go to **Navigation menu > Cloud Storage > Buckets** and click on the name of your bucket to see the results of the wordcount in the `wordcount` folder.

7. Once all the steps are complete in the DAG, each step has a dark green border. Additionally the Dataproc cluster that was created is now deleted.