# An Introduction to Cloud Composer 2.5

1 hour 30 minutes No cost

## 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

## Setup and requirements

### Lab setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time.
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

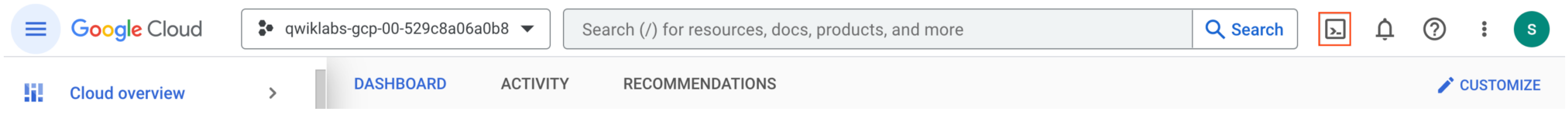
**Note:** Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

## Activate Google Cloud Shell

Google 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.

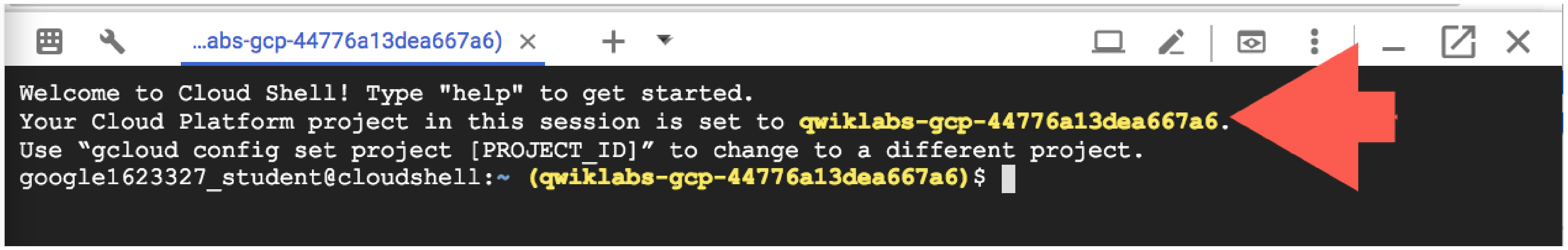
Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.


Google Cloud
qwiklabs-gcp-00-529c8a06a0b8
Search (/) for resources, docs, products, and more
Search
S
Cloud overview
DASHBOARD
ACTIVITY
RECOMMENDATIONS
CUSTOMIZE

2. 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:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

**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: - @.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 =

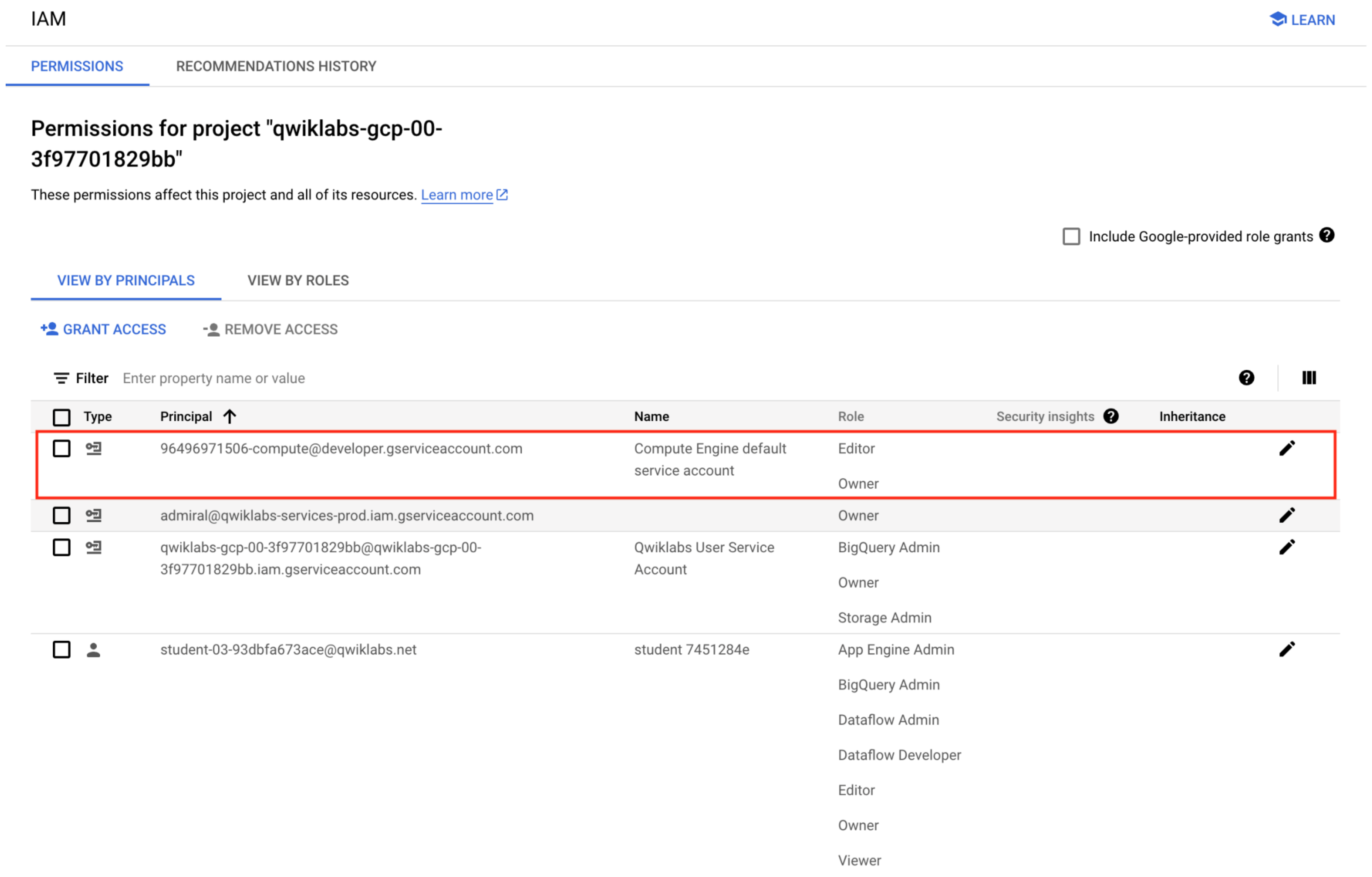
**Example output:**

[core] project = qwiklabs-gcp-44776a13dea667a6 **Note:** Full documentation of **gcloud** is available in the [gcloud CLI overview guide ](gcloud CLI overview guide).

## 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**.



Permissions for project "qwiklabs-gcp-00-3f97701829bb"

These permissions affect this project and all of its resources. Learn more

Include Google-provided role grants
VIEW BY PRINCIPALS
VIEW BY ROLES
GRANT ACCESS
REMOVE ACCESS
Filter
Enter property name or value

| Type | Principal | Name | Role | Security insights | Inheritance |
|---|---|---|---|---|---|
|  | 96496971506-compute@developer.gserviceaccount.com | Compute Engine default service account | Editor<br>Owner |  |  |
|  | admiral@qwiklabs-services-prod.iam.gserviceaccount.com |  | Owner |  |  |
|  | qwiklabs-gcp-00-3f97701829bb@qwiklabs-gcp-00-3f97701829bb.iam.gserviceaccount.com | Qwiklabs User Service Account | BigQuery Admin<br>Owner<br>Storage Admin |  |  |
|  | student-03-93dbfa673ace@qwiklabs.net | student 7451284e | App Engine Admin<br>BigQuery Admin<br>Dataflow Admin<br>Dataflow Developer<br>Editor<br>Owner<br>Viewer |  |  |

**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. Ensure that the Kubernetes Engine API is successfully enabled

To ensure access to the necessary APIs, restart the connection to the Kubernetes Engine API.

1. In the Google Cloud console, enter **Kubernetes Engine API** in the top search bar. Click on the result for **Kubernetes Engine API**.
2. Click **Manage**.
3. Click **Disable API**.

- If prompted to confirm, click **Disable**.
- If prompted again `Do you want to disable Kubernetes Engine API and its dependent APIs?`, click **Confirm**.

4. Click **Enable**.

When the API has been enabled again, the page will show the option to disable.

## Task 2. Ensure that the Cloud Composer API is successfully enabled

Restart the connection to the Cloud Composer API. In the prior step, restarting the Kubernetes Engine API forced the Cloud Composer API to be disabled.

1. In the Google Cloud console, enter **Cloud Composer API** in the top search bar. Click on the result for **Cloud Composer API**.
2. Click **Enable**.

When the API has been enabled, the page will show the option to disable.

## Task 3. 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. Go to **Navigation menu** > **Composer**.
2. Click **Create Environment** and select **Composer 1**. Set the following for your environment:

| Property | Value |
|---|---|
| **Name** | `highcpu` |
| **Location** | `us-central1` |
| **Zone** | `us-central1-a` |
| **Machine type** | e2-highcpu-4 |

Leave all other settings as default.

3. 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 10-20 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 Cloud Composer environment.

## 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, 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.

Create a Cloud Storage bucket.

## Task 4. 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 5. 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:

```
"""Example Airflow DAG that creates a Cloud Dataproc cluster, runs the Hadoop wordcount example, and deletes the cluster. This DAG relies on
three Airflow variables https://airflow.apache.org/concepts.html#variables * gcp_project - Google Cloud Project to use for the Cloud Dataproc
cluster. * gce_zone - Google Compute Engine zone where Cloud Dataproc cluster should be created. * gcs_bucket - Google Cloud Storage bucket to
used as output for the Hadoop jobs from Dataproc. See https://cloud.google.com/storage/docs/creating-buckets for creating a bucket. """ import
datetime import os from airflow import models from airflow.contrib.operators import dataproc_operator from airflow.utils import trigger_rule #
Output file for Cloud Dataproc job. output_file = os.path.join( models.Variable.get('gcs_bucket'), 'wordcount',
datetime.datetime.now().strftime('%Y%m%d-%H%M%S')) + os.sep # Path to Hadoop wordcount example available on every Dataproc cluster.
WORDCOUNT_JAR = ( 'file:///usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar' ) # Arguments to pass to Cloud Dataproc job.
wordcount_args = ['wordcount', 'gs://pub/shakespeare/rose.txt', output_file] yesterday = datetime.datetime.combine( datetime.datetime.today() -
datetime.timedelta(1), datetime.datetime.min.time()) default_dag_args = { # Setting start date as yesterday starts the DAG immediately when it is #
detected in the Cloud Storage bucket. 'start_date': yesterday, # To email on failure or retry set 'email' arg to your email and enable # emailing here.
'email_on_failure': False, 'email_on_retry': False, # If a task fails, retry it once after waiting at least 5 minutes 'retries': 1, 'retry_delay':
datetime.timedelta(minutes=5), 'project_id': models.Variable.get('gcp_project') } with models.DAG( 'composer_sample_quickstart', # Continue to run
DAG once per day schedule_interval=datetime.timedelta(days=1), default_args=default_dag_args) as dag: # Create a Cloud Dataproc cluster.
create_dataproc_cluster = dataproc_operator.DataprocClusterCreateOperator( task_id='create_dataproc_cluster', # Give the cluster a unique name by
appending the date scheduled. # See https://airflow.apache.org/code.html#default-variables cluster_name='composer-hadoop-tutorial-cluster-{{
ds_nodash }}', num_workers=2, region='us-central1', zone=models.Variable.get('gce_zone'), image_version='2.0', master_machine_type='n1-
standard-2', worker_machine_type='n1-standard-2') # Run the Hadoop wordcount example installed on the Cloud Dataproc cluster # master node.
run_dataproc_hadoop = dataproc_operator.DataProcHadoopOperator( task_id='run_dataproc_hadoop', region='us-central1',
main_jar=WORDCOUNT_JAR, cluster_name='composer-hadoop-tutorial-cluster-{{ ds_nodash }}', arguments=wordcount_args) # Delete Cloud
Dataproc cluster. delete_dataproc_cluster = dataproc_operator.DataprocClusterDeleteOperator( task_id='delete_dataproc_cluster', region='us-
central1', cluster_name='composer-hadoop-tutorial-cluster-{{ ds_nodash }}', # Setting trigger_rule to ALL_DONE causes the cluster to be deleted #
even if the Dataproc job fails. trigger_rule=trigger_rule.TriggerRule.ALL_DONE) # Define DAG dependencies. create_dataproc_cluster >>
run_dataproc_hadoop >> delete_dataproc_cluster
```

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 `quickstart`, and the DAG runs once each day:

with models.DAG( 'composer_sample_quickstart', # 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 6. 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 7. Using the Airflow UI

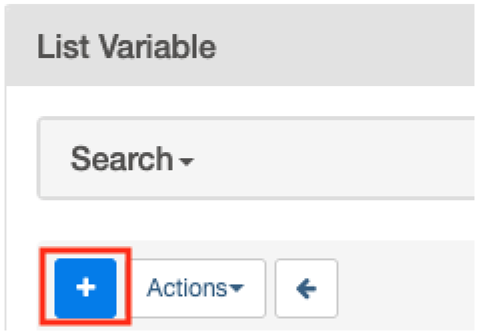
To access the Airflow web interface using the console:

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 8. Setting Airflow variables

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

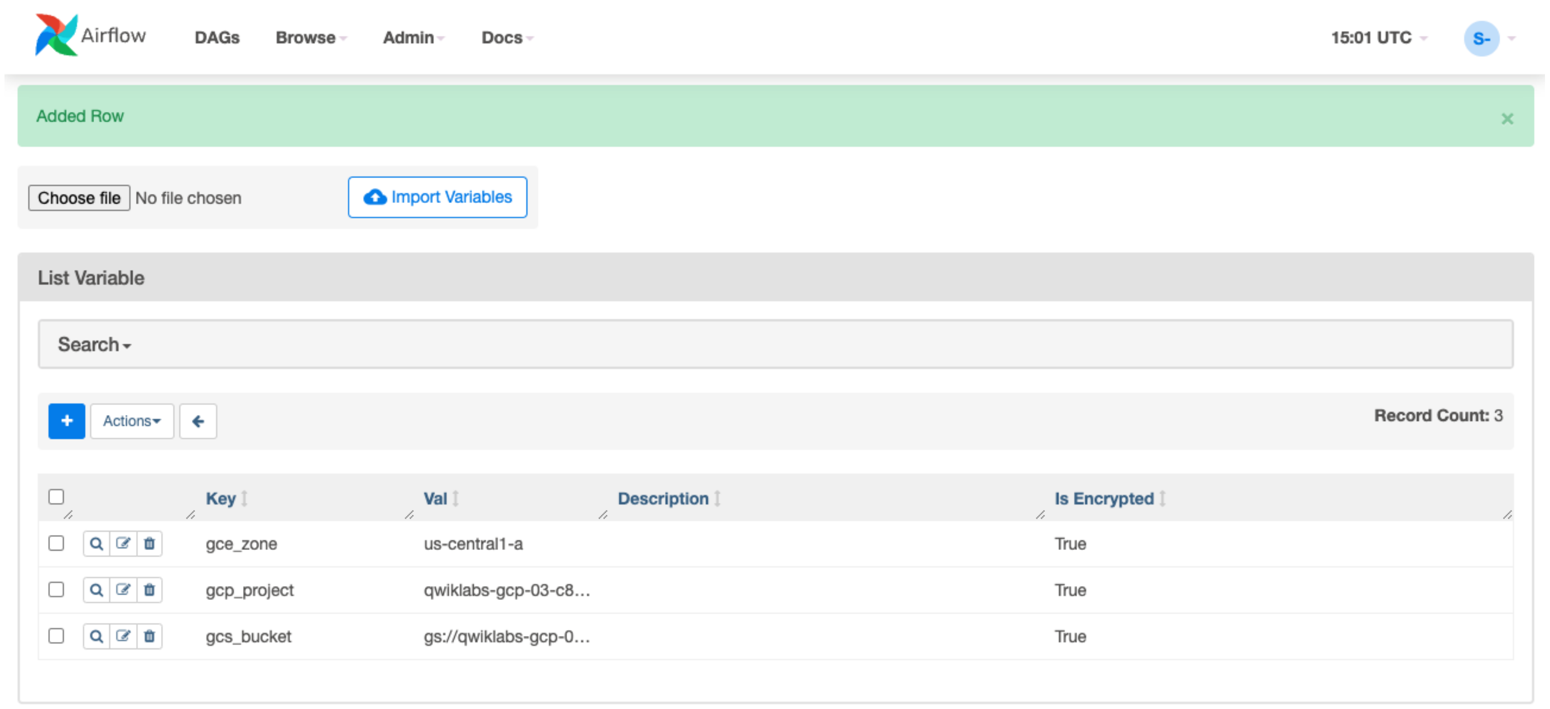
1. From the Airflow interface, select **Admin** > **Variables** from the menu bar.
2. Click **+** icon to add a new record.



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

| Key | Val | Details |
|---|---|---|
| `gcp_project` | <your project-id> | The Google Cloud Platform project you're using for this quickstart. |
| `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 from Dataproc. |
| `gce_zone` | `us-central1-a` | This is the Compute Engine zone where your Cloud Dataproc cluster will be created. To choose a different zone, see Available regions & zones. |

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:



Added Row

List Variable

Record Count: 3

| Key | Val | Description | Is Encrypted |
|---|---|---|---|
| gce_zone | us-central1-a | | True |
| gcp_project | qwiklabs-gcp-03-c8… | | True |
| gcs_bucket | gs://qwiklabs-gcp-0… | | True |

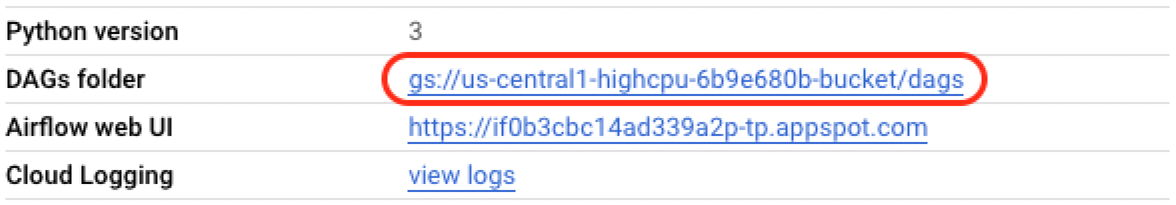
## Task 9. Uploading the DAG to Cloud Storage

To upload the DAG:

1. In 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.

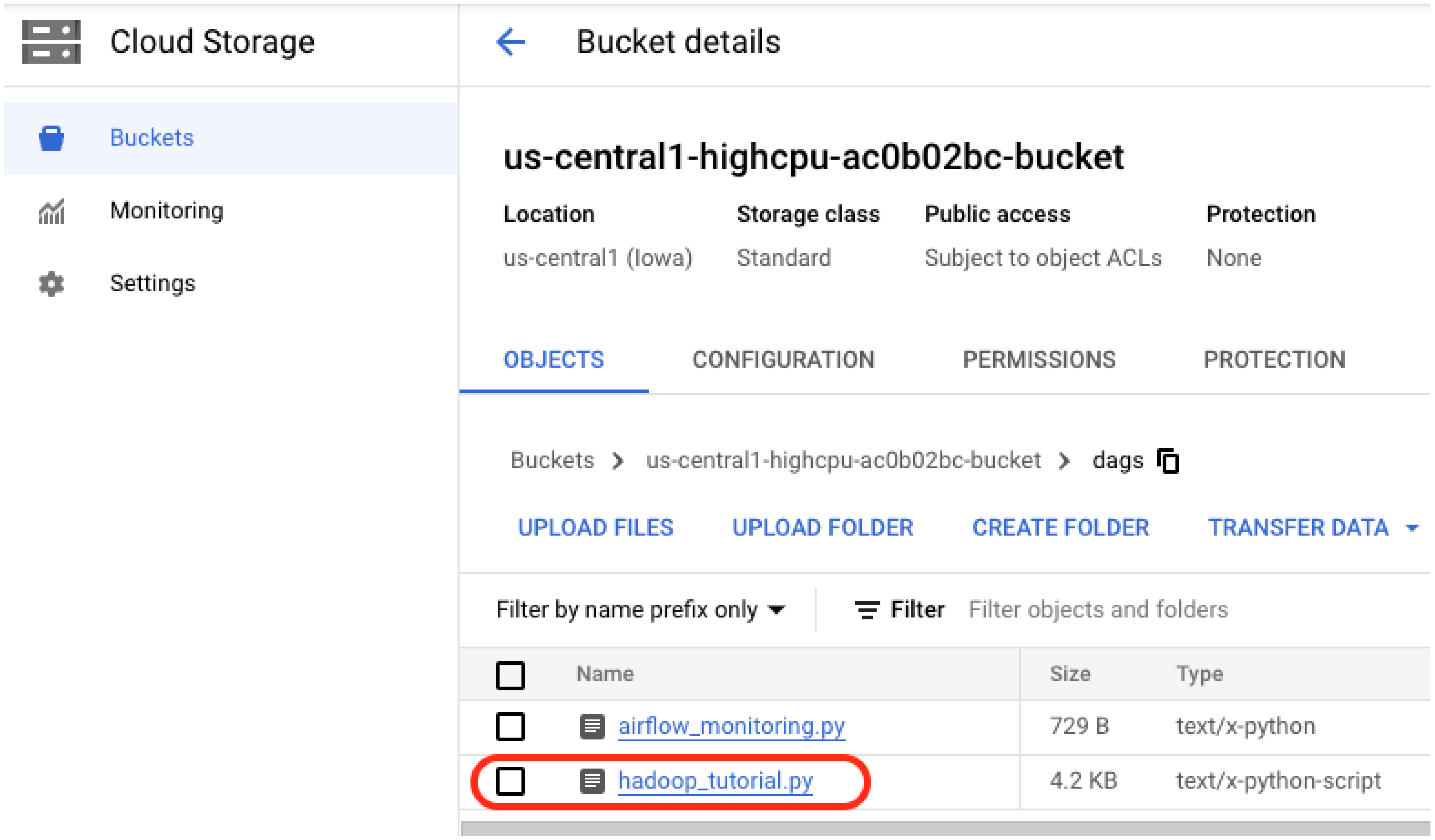


| | |
|---|---|
| Python version | 3 |
| DAGs folder | gs://us-central1-highcpu-6b9e680b-bucket/dags |
| Airflow web UI | https://if0b3cbc14ad339a2p-tp.appspot.com |
| Cloud Logging | view logs |

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-central1-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 |

OBJECTS
CONFIGURATION
PERMISSIONS
PROTECTION
Buckets > us-central1-highcpu-ac0b02bc-bucket > dags
UPLOAD FILES
UPLOAD FOLDER
CREATE FOLDER
TRANSFER DATA
Filter by name prefix only
Filter
Filter objects and folders

| 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.

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

Uploading the DAG 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.

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.

## Congratulations!

You've successfully run a Cloud Composer workflow!

## Next steps

- Check out when Cloud Composer was presented at NEXT 18 in San Francisco: Flexible, Easy Data Pipelines on Google Cloud with Cloud Composer (Cloud Next '18)
- To see the value of a variable, run the Airflow CLI sub-command variables with the get argument or use the Airflow web interface.
- For information about the Airflow web interface, see Accessing the web interface.

## End your lab

When you have completed your lab, click **End Lab**

-