An Introduction to Cloud Composer | Qwiklabs

Saturday, November 14, 2020 3:41 PM

Clipped from: https://googlecourses.qwiklabs.com/course_sessions/71782/labs/11637

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 Platform (GCP), 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 GCP 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 GCP 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

Qwiklabs setup

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

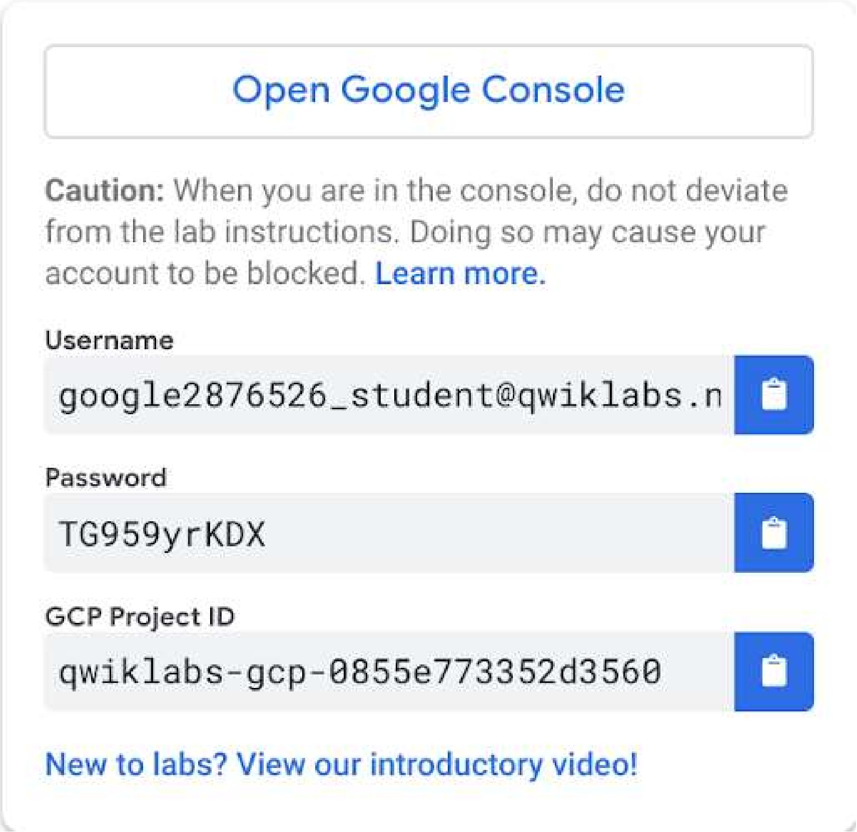
1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example,

02:00:00

and make sure you can finish in that time block.
3. When ready, click

.

4. Note your lab credentials. You will use them to sign in to the Google Cloud Console.



Open Google Console

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

google2876526_student@qwiklabs.n

Password

TG959yrKDX

GCP Project ID

qwiklabs-gcp-0855e773352d3560

New to labs? View our introductory video!

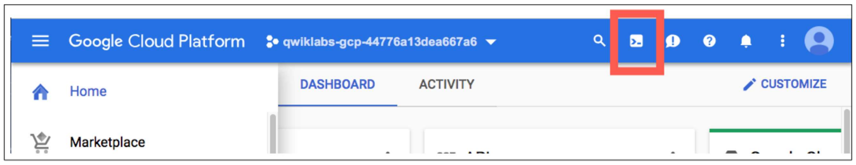
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

1. Accept the terms and skip the recovery resource page.

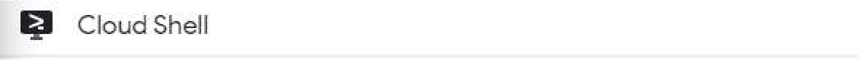
Activate Cloud Shell

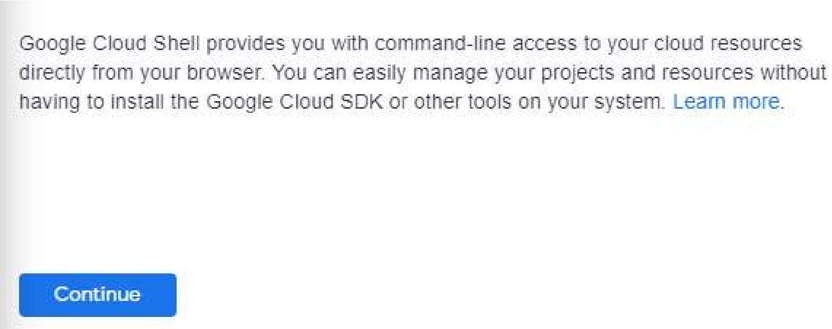
Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

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



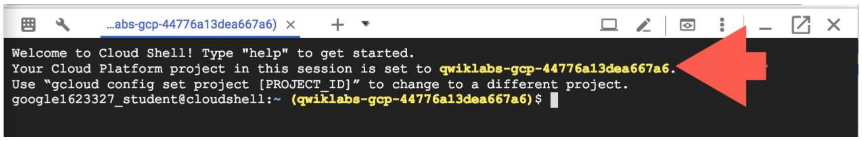
Click **Continue**.





Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

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:
- <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>
```

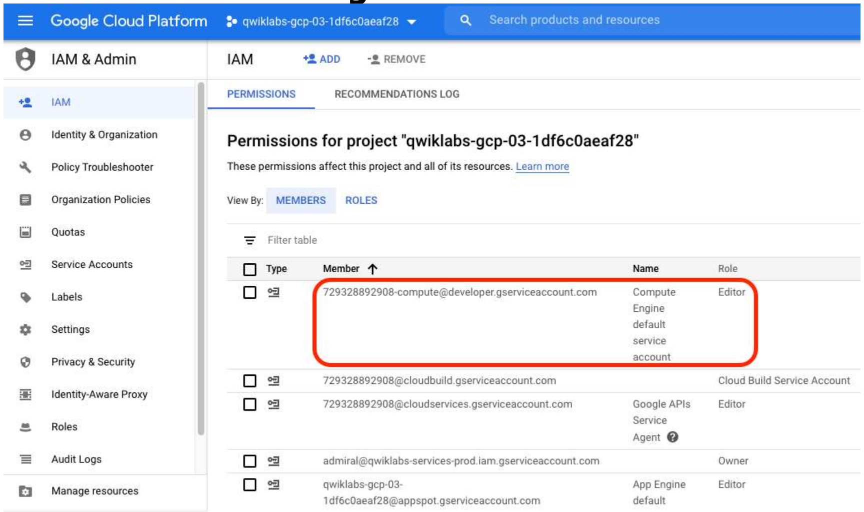
(Example output)

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

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**.



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

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

Type	Member	Name	Role
□	729328892908-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor
□	729328892908@cloudbuild.gserviceaccount.com		Cloud Build Service Account
□	729328892908@cloudservices.gserviceaccount.com	Google APIs Service Agent	Editor
□	admiral@qwiklabs-services-prod.iam.gserviceaccount.com		Owner
□	qwiklabs-gcp-03-1df6c0aeaf28@appspot.gserviceaccount.com	App Engine default	Editor

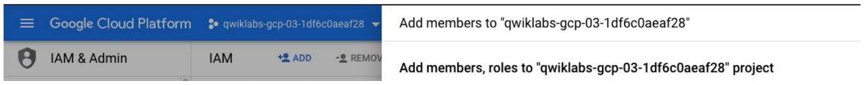
If the account is not present in IAM or does not have the `editor` role, follow the steps below to assign the required role.

- 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`

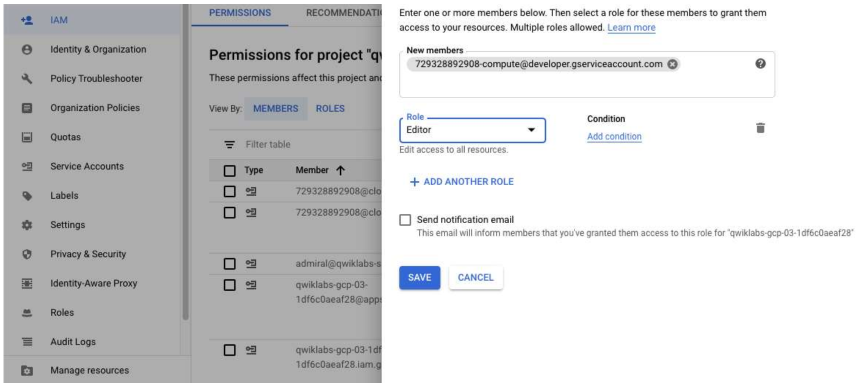
Replace `{project-number}` with your project number.

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



Add members to "qwiklabs-gcp-03-1df6c0aeaf28"

Add members, roles to "qwiklabs-gcp-03-1df6c0aeaf28" project



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

New members: 729328892908-compute@developer.gserviceaccount.com

Role: Editor

Edit access to all resources.

Condition: Add condition

+ ADD ANOTHER ROLE

□ Send notification email

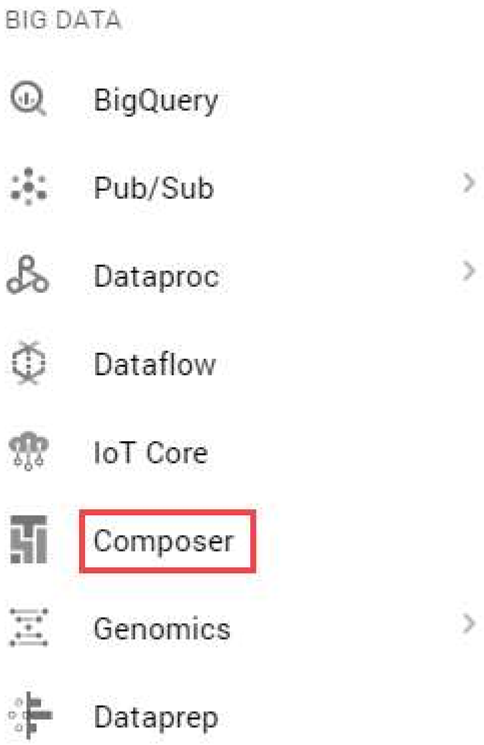
This email will inform members that you've granted them access to this role for "qwiklabs-gcp-03-1df6c0aeaf28"

SAVE CANCEL

Create Cloud Composer environment

In this section, you create a Cloud Composer environment.

1. Go to **Navigation menu** > **Composer**:



1. Click **CREATE ENVIRONMENT** and set the following for your environment:

Property	**Value**
Name	`highcpu`
Location	`us-central1`
Zone	`us-central1-a`
Machine type	n1-highcpu-4

Leave all other settings as default.

1. 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 GCP Console.

It can take 10-15 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 buckets will be used as output for the Hadoop job from Dataproc.

1. Go to **Navigation menu** > **Storage** > **Browser** and then click **Create bucket**.
2. Give your bucket a universally unique name, then click **Create**.

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

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

Create a Cloud Storage bucket.

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 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*, ...

You can read more about the concepts here.

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='quickstart-cluster-{{
ds_nodash }}',
        num_workers=2,
        zone=models.Variable.get('gce_zone'),
        master_machine_type='n1-standard-1',
        worker_machine_type='n1-standard-1')

    # Run the Hadoop wordcount example installed
on the Cloud Dataproc cluster
    # master node.
    run_dataproc_hadoop =
dataproc_operator.DataProcHadoopOperator(
        task_id='run_dataproc_hadoop',
        main_jar=WORDCOUNT_JAR,
        cluster_name='quickstart-cluster-{{
ds_nodash }}',
        arguments=wordcount_args)

    # Delete Cloud Dataproc cluster.
    delete_dataproc_cluster =
dataproc_operator.DataprocClusterDeleteOperator(
        task_id='delete_dataproc_cluster',
        cluster_name='quickstart-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.

Viewing environment information

1. Go back to **Composer** to check on 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 details** you'll see information such as the Airflow web interface URL, Kubernetes Engine cluster ID, and a link to the DAGs folder, which is stored in your bucket.

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

Using the Airflow UI

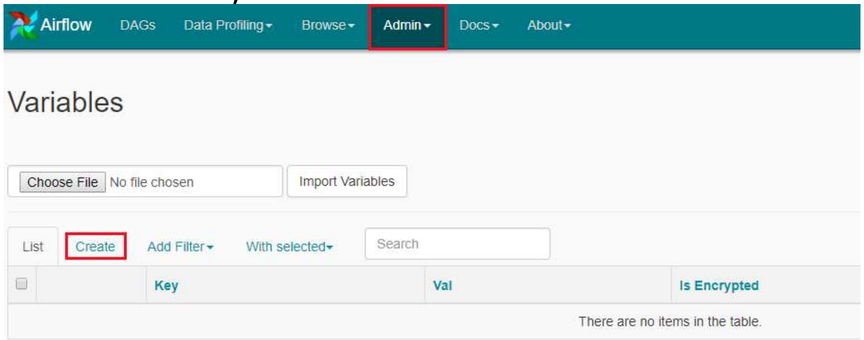
To access the Airflow web interface using the GCP Console:

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.

Setting Airflow variables

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

1. Select **Admin** > **Variables** from the Airflow menu bar, then **Create**.



Variables

Key	Val	Is Encrypted

There are no items in the table.

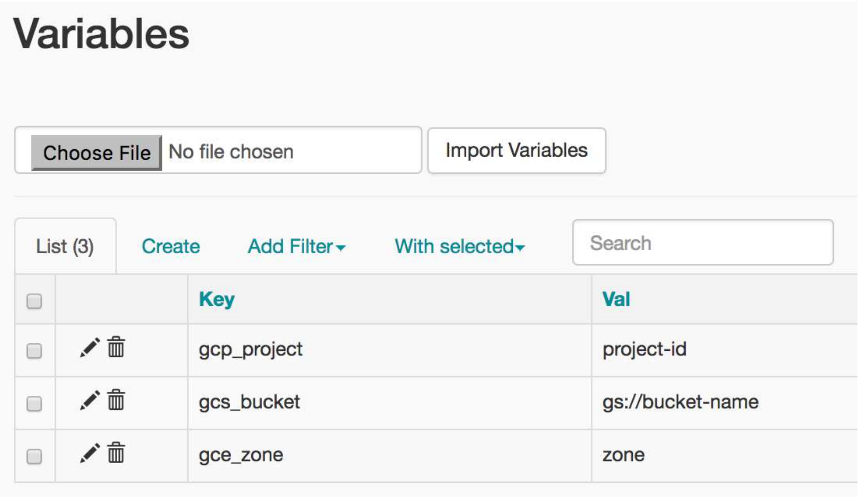
1. Create the following Airflow variables, `gcp_project`, `gcs_bucket`, and `gce_zone`:

KEYVALUEDetails

`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 chose a different zone, see Available regions & zones.

Click **Save and Add Another** after adding first two variable and click **Save** for the third variable. Your

Variables table should look like this when you're finished:



Variables

☐		Key	Val
☐	✎ 🗑	gcp_project	project-id
☐	✎ 🗑	gcs_bucket	gs://bucket-name
☐	✎ 🗑	gce_zone	zone

Uploading the DAG to Cloud Storage

To upload the DAG:

1. In Cloud Shell, copy and save [hadoop_tutorial.py](#) on your local virtual machine:

```
git clone
https://github.com/GoogleCloudPlatform/python-docs-samples
```

1. Change to the `python-docs-samples` directory:

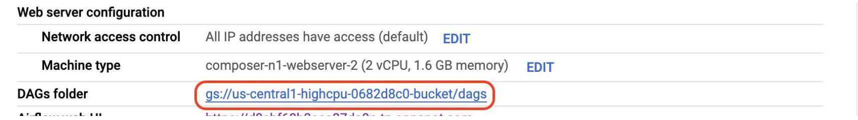
```
cd python-docs-samples/composer/workflows
```

1. Now upload a copy of the `hadoop_tutorial.py` file to the Cloud Storage bucket that was automatically created when you created the environment.

To upload the file using Cloud Shell, replace `<DAGs_folder_path>` in the following command:

```
gsutil cp hadoop_tutorial.py <DAGs_folder_path>
```

with the path to the DAGs folder. 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.



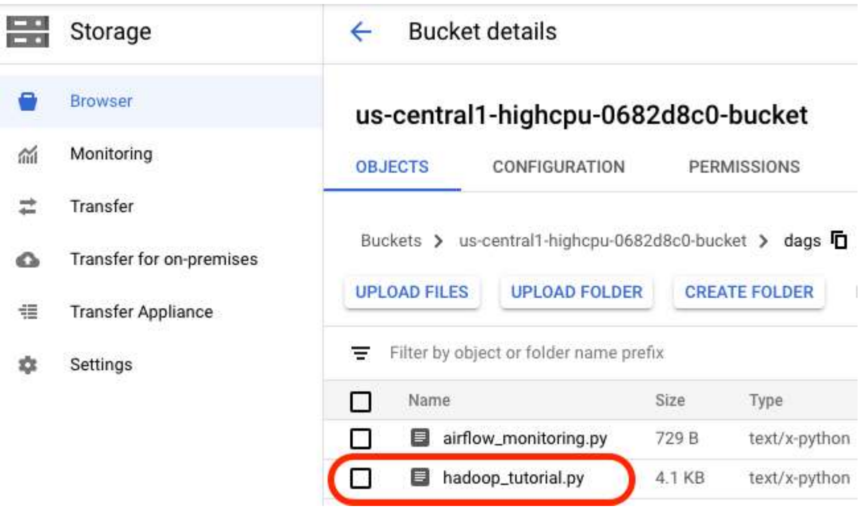
Web server configuration		
Network access control	All IP addresses have access (default)	EDIT
Machine type	composer-n1-webserver-2 (2 vCPU, 1.6 GB memory)	EDIT
DAGs folder	gs://us-central1-highcpu-0682d8c0-bucket/dags	

Airflow web UI	https://duebf68b3eaa37daup-tp.appspot.com

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

```
gsutil cp hadoop_tutorial.py gs://us-central1-highcpu-0682d8c0-bucket/dags
```

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-0682d8c0-bucket

Buckets > us-central1-highcpu-0682d8c0-bucket > dags

	Name	Size	Type
□	airflow_monitoring.py	729 B	text/x-python
□	hadoop_tutorial.py	4.1 KB	text/x-python

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.

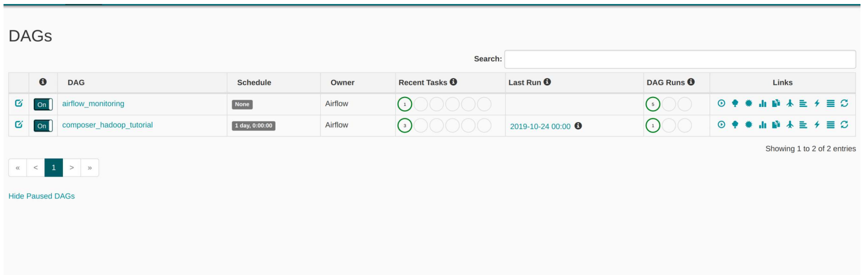
Click **Check my progress** to verify the objective.

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.

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.



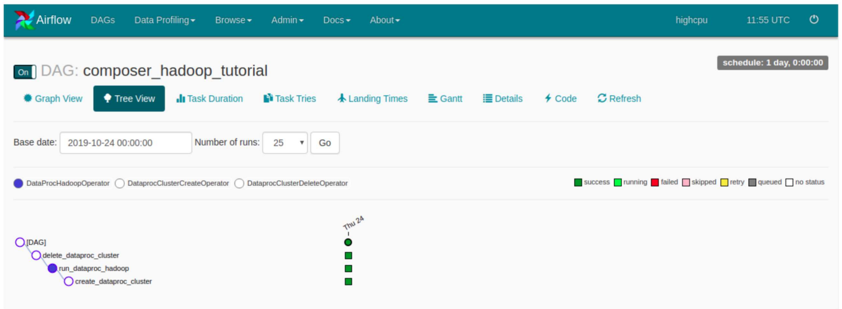
DAGs

Search:

	ℹ	DAG	Schedule	Owner	Recent Tasks ℹ	Last Run ℹ	DAG Runs ℹ	Links
	On	airflow_monitoring	None	Airflow	1		5	
	On	composer_hadoop_tutorial	1 day, 0:00:00	Airflow	3	2019-10-24 00:00 ℹ	1	

Showing 1 to 2 of 2 entries

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

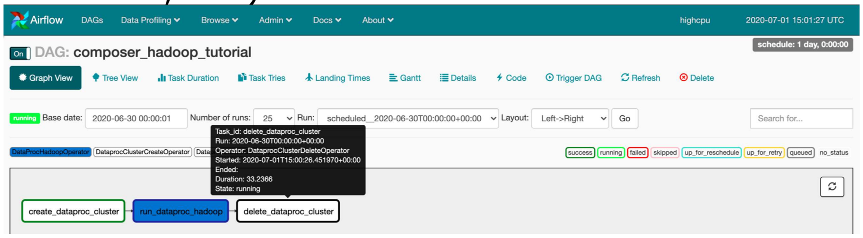


DAG: composer_hadoop_tutorial

schedule: 1 day, 0:00:00

Base date: 2019-10-24 00:00:00 Number of runs: 25

1. In the toolbar, click **Graph View**. 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.).



DAG: composer_hadoop_tutorial

schedule: 1 day, 0:00:00

Base date: 2020-06-30 00:00:01 Number of runs: 25 Run: scheduled__2020-06-30T00:00:00+00:00 Layout: Left->Right

```
Task_id: delete_dataproc_cluster
Run: 2020-06-30T00:00:00+00:00
Operator: DataprocClusterDeleteOperator
Started: 2020-07-01T15:00:26.451970+00:00
Ended:
Duration: 33.2366
State: running
```

1. Click the "Refresh" link to make sure you're looking at the most recent information. The borders of the processes change colors as the state of the process changes
2. 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.

1. 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** > **Storage** > **Browser** and click on the name of your bucket to see the results of the wordcount in the `wordcount` folder.

1. 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: https://www.youtube.com/watch?v=GeNFEtt-D4k
- 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**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.