

Cloud Composer: Qwik Start - Console

 cloudskillsboost.google/games/2854/labs/17208

GSP261



Google Cloud Self-Paced Labs

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 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 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

Qwiklabs 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 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.

To complete this lab, you need:

Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

Time to complete the lab---remember, once you start, you cannot pause a lab.

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

Cloud Console

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 the **Lab Details** panel with the following:
 - The **Open Google Console** button
 - Time remaining
 - The temporary credentials that you must use for this lab
 - Other information, if needed, to step through this lab
2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

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

Note: If you see the **Choose an account** dialog, click **Use Another Account**.

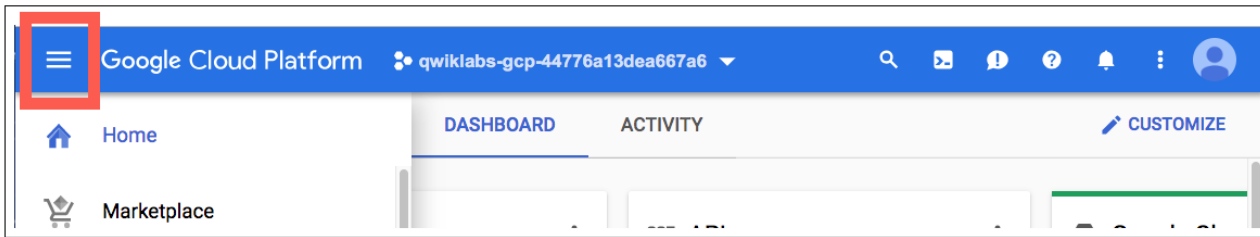
3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.

5. 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.

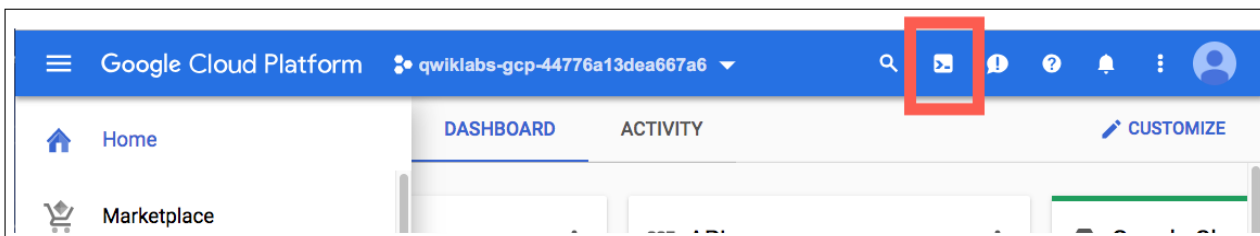
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



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.

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



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**. The output contains a line that declares the **PROJECT_ID** for this session:

Your Cloud Platform project in this session is set to YOUR_PROJECT_ID
`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

(Output)

ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxxx@qwiklabs.net To set the active account, run: `$ gcloud config set account `ACCOUNT``

4. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

(Output)

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

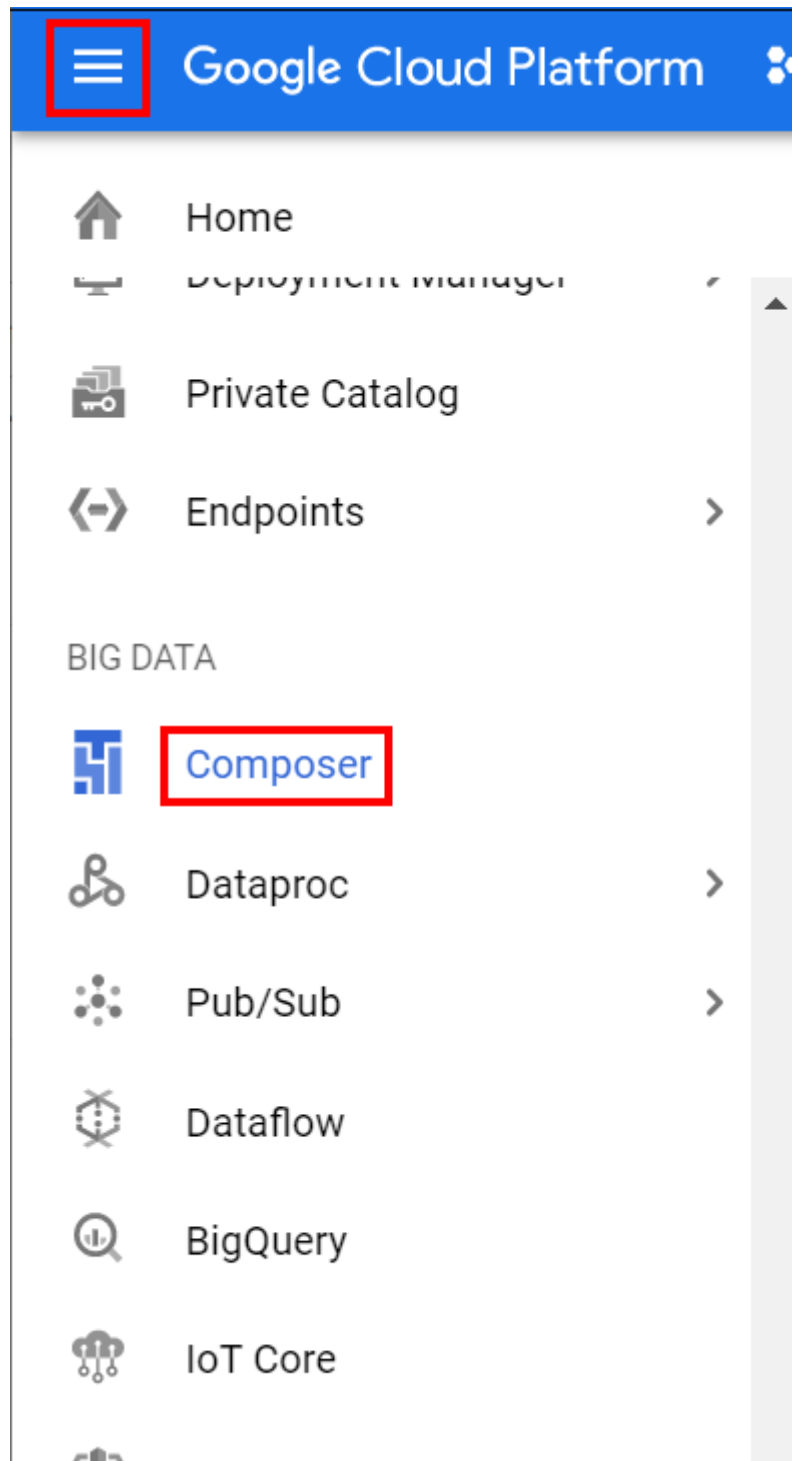
(Example output)

[core] project = qwiklabs-gcp-44776a13dea667a6 For full documentation of `gcloud`, in Google Cloud, Cloud SDK documentation, see the [gcloud command-line tool overview](#).

Create Cloud Composer environment

In this section, you create a Cloud Composer environment.

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



2. Click **CREATE ENVIRONMENT** and select **Composer 1** from the dropdown. Set the following for your environment:

Name: highcpu

Location: us-central1

Zone: us-central1-a

Machine type: n1-highcpu-4

Image version: select latest version

Python version: 3

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 Cloud Console.

It can take 10-15 minutes for the environment to complete the setup process. Continue with the lab while the environment spins up.

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 > Cloud 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 code for the `hadoop_tutorial.py` workflow, 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
use for result of Hadoop job. 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') } # [START composer_hadoop_schedule] 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:
# [END composer_hadoop_schedule] # 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) # [START
composer_hadoop_steps] # Define DAG dependencies. create_dataproc_cluster >>
run_dataproc_hadoop >> delete_dataproc_cluster # [END composer_hadoop_steps]
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 **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.

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.

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

Create Cloud Composer environment.

Using the Airflow UI

To access the Airflow web interface using the Cloud 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 click + (**Add a new record**).

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

KEY	VALUE	Details
<code>gcp_project</code>	your project-id	The Google Cloud project you're using for this quickstart.
<code>gcs_bucket</code>	gs://	Replace with the name of the Cloud Storage bucket you made earlier. This bucket stores the output from the Hadoop jobs from Dataproc.
<code>gce_zone</code>	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 .

Note: After entering a record click **Save** and then again click on + to enter a new record.

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

	Key	Val	Description	Is Encrypted
<input type="checkbox"/>	gce_zone	us-central1-a		True
<input type="checkbox"/>	gcp_project	qwiklabs-gcp-00-b77618186648		True
<input type="checkbox"/>	gcs_bucket	gs://qwiklabs-gcp-00-b77618186648		True

Uploading the DAG to Cloud Storage

To upload the DAG:

1. In Cloud Shell, upload a copy of the `hadoop_tutorial.py` file to the Cloud Storage bucket that was automatically created when you created the environment. You can check that by going to **Composer > Environments**. Click on the environment you created earlier, this will get you to the description of the environment you created. Find **DAGs folder**, copy the value to replace `<DAGs_folder_path>` in the following command to upload the file :

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

Cloud Composer adds the DAG to Airflow and schedules the DAG automatically. DAG changes occur within 3-5 minutes. The workflow will now be referred to as `composer_hadoop_tutorial`.

You will be able to see the task status in the Airflow web interface.

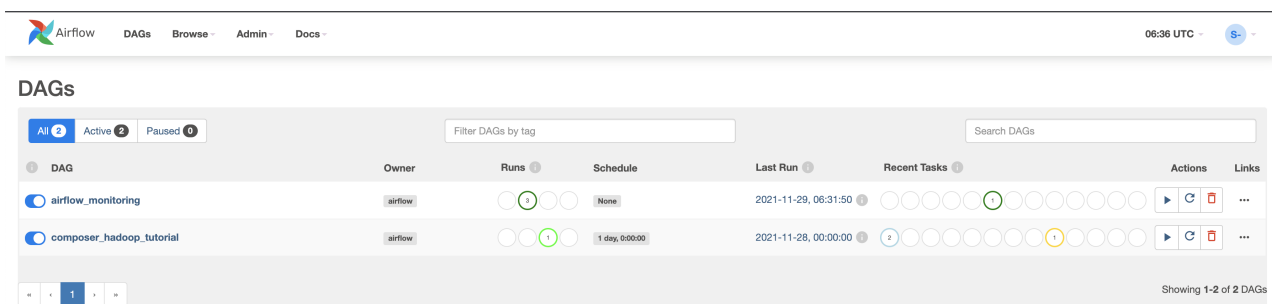
Click **Check my progress** to verify the objective.

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.

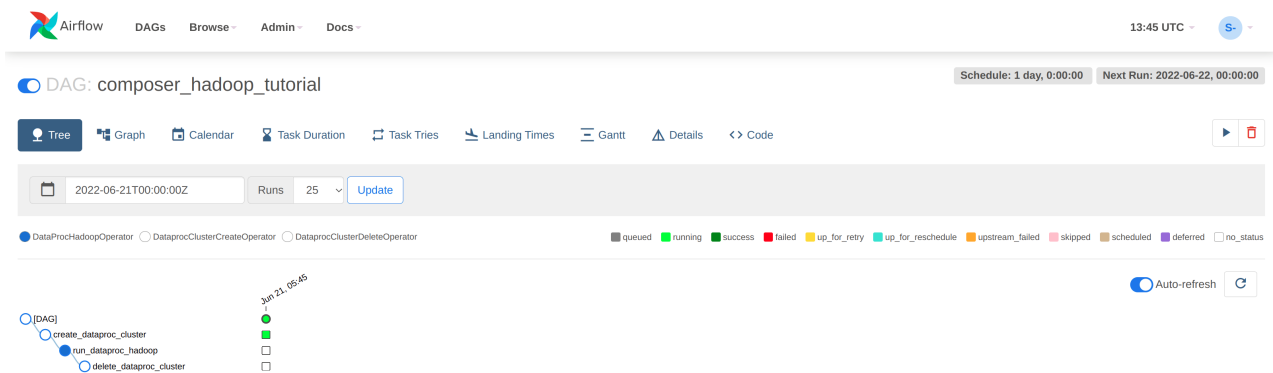
Make sure that you're on the DAGs tab in the Airflow web interface. It takes several minutes for this process to complete. Refresh your browser to make sure you're looking at the latest information.



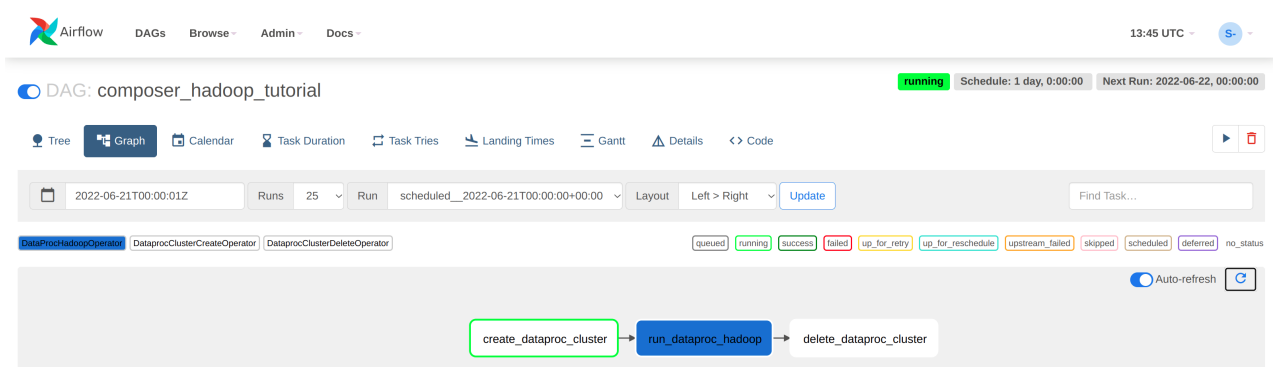
The screenshot shows the Airflow web interface with the 'DAGs' tab selected. The interface includes a navigation bar with 'Airflow', 'DAGs', 'Browse', 'Admin', and 'Docs'. The top right shows the time '06:36 UTC' and a user icon. Below the navigation bar, there's a 'DAGs' section with filters for 'All' (2), 'Active' (2), and 'Paused' (0). A search bar 'Filter DAGs by tag' and 'Search DAGs' are present. The main table lists DAGs with columns: DAG, Owner, Runs, Schedule, Last Run, Recent Tasks, Actions, and Links. Two DAGs are listed: 'airflow_monitoring' and 'composer_hadoop_tutorial'. The 'composer_hadoop_tutorial' DAG is highlighted, showing its last run on 2021-11-28 at 00:00:00 and a recent task status of '1'.

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links
airflow_monitoring	airflow	3	None	2021-11-29, 06:31:50	1	[Icons]	...
composer_hadoop_tutorial	airflow	1	1 day, 0:00:00	2021-11-28, 00:00:00	1	[Icons]	...

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



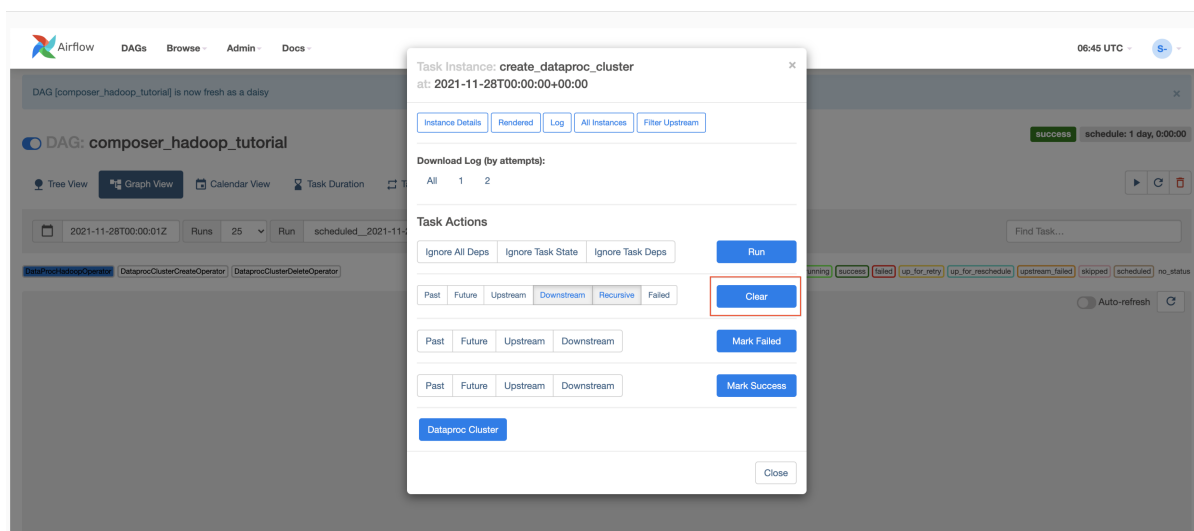
2. In the toolbar, click on **Graph** for graph view. Mouseover the graphic for each task to see its status. Note that the border around each task also indicates the status (green border = running; red = failed, etc.).



3. If required, turn on **Auto refresh** or click the **Refresh** icon 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.

Once your process reaches the Success state, run the workflow again from the **Graph** view:

1. Click the **create_dataproc_cluster** graphic.
2. Click **Clear** to reset the three tasks.



3. Then click **OK** to confirm.

Notice that the color around **create_dataproc_cluster** has changed and the state is "running".

You can also monitor the process in the Cloud Console.

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

Test your knowledge

Test your knowledge about Google cloud Platform by taking our quiz.

Delete Cloud Composer Environment

1. Return to the **Environments** page in **Composer**.
2. Select the checkbox next to your **Composer** environment.
3. Click **DELETE**.
4. Confirm the pop-up by clicking **DELETE** again.

Congratulations!

You've taken your first steps into a larger world of deep learning!

Next steps

This lab is part of a series of labs called Qwik Starts. These labs are designed to give you a little taste of the many features available with Google Cloud. Search for "Qwik Starts" in the [lab catalog](#) to find the next lab you'd like to take!

- 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.

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 June 22, 2022

Lab Last Tested June 22, 2022

Copyright 2022 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.