Continuous Training Pipelines with Cloud Composer

2 hours Free

Overview

In this lab you will learn how to write an Airflow DAG for continuous training and deploy the DAG within a Cloud Composer environment. You will also learn how to explore and monitor your DAG runs using the Apache Airflow webserver.

Objectives

In this lab, you will learn to perform the following tasks:

- Provision a Cloud Composer environment.
- Deploy an Apache Airflow Dialog.
- Monitor a continuous training pipeline in the Airflow webserver.
- Explore Airflow logs using Cloud Operations.

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.

Log in to Google Cloud Console

1. Using the browser tab or window you are using for this lab session, copy the **Username** from the **Connection Details** panel and click the **Open Google Console** button.

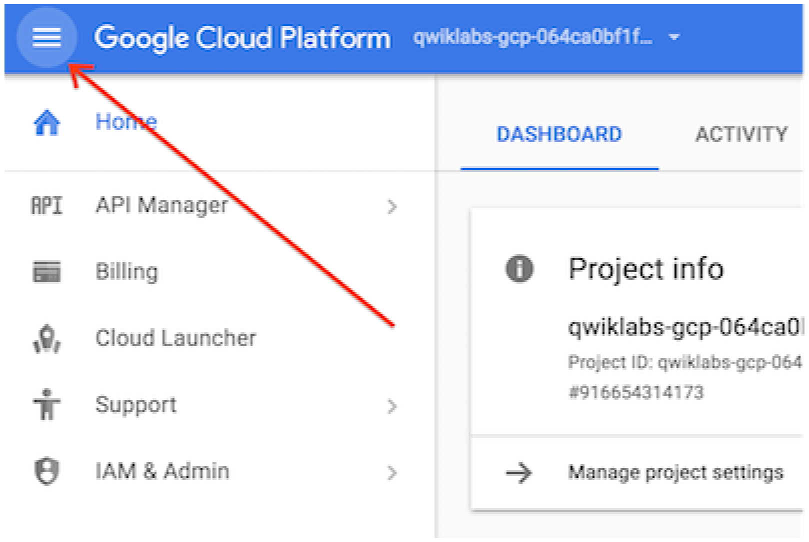
Note: If you are asked to choose an account, click **Use another account**.

2. Paste in the **Username**, and then the **Password** as prompted.
3. Click **Next**.
4. Accept the terms and conditions.

Since this is a temporary account, which will last only as long as this lab:

- Do not add recovery options
- Do not sign up for free trials

5. Once the console opens, view the list of services by clicking the **Navigation menu** (☰) at the top-left.



Activate Cloud Shell

Cloud Shell is a virtual machine that contains development tools. It offers a persistent 5-GB home directory and runs on Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources. `gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab completion.

1. Click the **Activate Cloud Shell** button ( ) at the top right of the console.
2. Click **Continue**.
   It takes a few moments to provision and connect to the environment. When you are connected, you are also authenticated, and the project is set to your *PROJECT_ID*.

Sample commands

- List the active account name:

gcloud auth list

(Output)

Credentialed accounts: - <myaccount>@<mydomain>.com (active)

(Example output)

Credentialed accounts: - google1623327_student@qwiklabs.net

- List the project ID:

```
gcloud config list project
```

(Output)

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

(Example output)

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

Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide.

Ensure that the Cloud Composer API is successfully enabled

To ensure access to the necessary APIs, restart the connection to the Cloud Composer API.

1. In the Google Cloud Console, enter **Cloud Composer API** in the top search bar.
2. Click on the result for **Cloud Composer API**.
3. Click **Manage**.
4. Click **Disable API**.

If asked to confirm, click **Disable**.

5. Click **Enable**.

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

Task 1. Provision Google Cloud resources

In this task, you configure and deploy a Cloud Composer environment to run a fully-managed installation of Apache Airflow.

Enabling Cloud service

- Run the following command to enable the required Cloud service:

```
gcloud services enable ml.googleapis.com
```

Download code repository

- In the Cloud Shell terminal, run the following commands:

```
git clone https://github.com/GoogleCloudPlatform/mlops-on-gcp cd mlops-on-gcp/continuous_training/composer/labs/ ls
```

You should see a directory `chicago_taxifare`, a JSON file `vars.json`, and a Python script `chicago_taxi_dag.py`.

Provision Google Cloud resources and stage trainer package

1. In the Cloud Shell terminal, run the following command to create a regional Cloud Storage bucket in the us-central1 region:

```
export BUCKET_NAME=${DEVSHELL_PROJECT_ID} export REGION=us-central1 export ZONE=us-central1-a gsutil mb -l ${REGION}
gs://${BUCKET_NAME}
```

2. Compress the trainer package and copy to your Cloud Storage bucket by running the following commands:

```
tar -cvf trainer.tar chicago_taxifare gsutil cp ./trainer.tar gs://${BUCKET_NAME}/chicago_taxi/code/
```

3. Create a Pub/Sub topic to receive messages from your Pub/Sub pipeline by running the following command:

```
gcloud pubsub topics create chicago-taxi-pipeline
```

4. Create a BigQuery dataset for storing preprocessed data for training and a table for storing training metrics by running the following commands:

```
bq mk -d chicago_taxi_ct bq mk --table chicago_taxi_ct.model_metrics version_name:STRING,rmse:FLOAT
```

Create Cloud Composer environment

- Create your Cloud Composer environment by running the following command in Cloud Shell:

  ```
  gcloud composer environments create demo-environment \ --location $REGION \ --zone $ZONE \ --python-version 3 \ --image-version
  composer-1.20.8-airflow-1.10.15
  ```

It will take 10-15 minutes to create the environment in Cloud Composer. Continue on to the next task while you are waiting.

Click *Check my progress* to verify the objective. Provision Google Cloud Resources

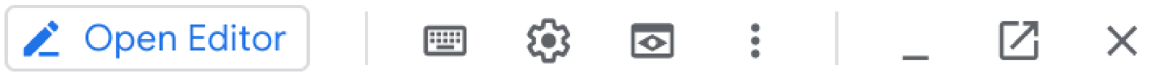
Task 2. Write DAG in Apache Airflow

In the following exercises you will perform a walkthrough of the Python script defining the Airflow DAG for our continuous training pipeline that we will later deploy to Cloud Composer. There are a few TODOs in the script that you will be completing along the way.

Launch Google Cloud Shell Code Editor

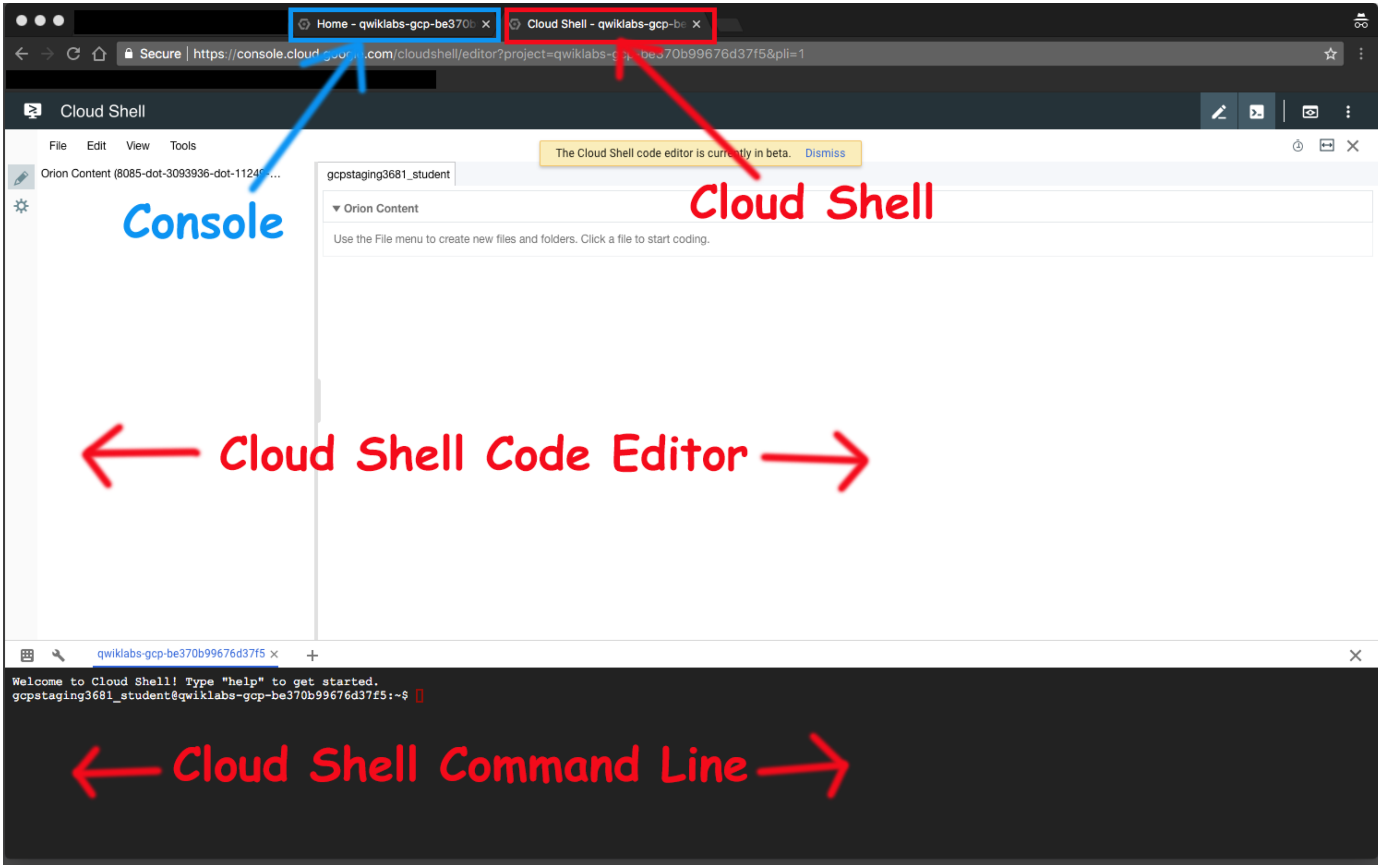
Use the Google Cloud Shell Code Editor to easily create and edit directories and files in the Cloud Shell instance.

Once you activate the Google Cloud Shell, click the **Open editor** button to open the Cloud Shell Code Editor.



You now have three interfaces available:

- The Cloud Shell Code Editor
- The console (by clicking on the tab). You can switch back and forth between the console and Cloud Shell by clicking on the tab
- The Cloud Shell command line (by clicking on **Open Terminal** in the console)


Home - qwiklabs-gcp-be370b
Cloud Shell - qwiklabs-gcp-be
Secure | https://console.cloud.google.com/cloudshell/editor?project=qwiklabs-gcp-be370b99676d37f5&pli=1
Cloud Shell
File Edit View Tools
The Cloud Shell code editor is currently in beta. Dismiss
Orion Content (8085-dot-3093936-dot-1124...
gcpstaging3681_student
Orion Content
Use the File menu to create new files and folders. Click a file to start coding.
Console
Cloud Shell
Cloud Shell Code Editor
qwiklabs-gcp-be370b99676d37f5
Cloud Shell Command Line

```
Welcome to Cloud Shell! Type "help" to get started.
gcpstaging3681_student@qwiklabs-gcp-be370b99676d37f5:~$
```

Navigate to DAG script

- Nagivate to `mlops-on-gcp/continuous_training/composer/labs` and open `chicago_taxi_dag.py`

We will be working through the Python script defining our continuous training DAG in Airflow. There will be five `#TODO` statements in the script which will specify tasks you will need to accomplish. For those who wish to go through the script for the DAG more carefully, detailed comments are included in the code.

If you get stuck, a solution (the completed `chicago_taxi_dag.py` script) can be found in the `mlops-on-gcp/continuous_training/composer/solutions` directory.

The Apache Airflow Concepts and Google Cloud Operators documentation may be helpful in the following exercises.

Explore and complete the DAG script

TODO 1: Complete the following code (starting around line 92) to instantiate the DAG.

with DAG(#ADD YOUR CODE HERE) as dag:

- Set the name of the DAG to be `chicago_taxi_dag`
- Pass in `DEFAULT_ARGS` as the default arguments
- Schedule the DAG to run weekly
- Setup the `catchup` argument to `False`

TODO 2: Next we define SQL queries for extracting the training and validation datasets. Starting at about line 175. Complete the following task definitions to execute these queries and save the results into new tables.

For `bq_train_data_op`:

- Set the `task_id` to `bq_train_data_task`
- Use the `bql_train` SQL query for the task
- Ensure that the task will use Standard SQL, not Legacy SQL
- Assign the DAG `dag` to the task

bq_train_data_op = BigQueryOperator(destination_dataset_table="{}.{}_train_data" .format(DESTINATION_DATASET, model.replace(".", "_")), write_disposition="WRITE_TRUNCATE", # specify to truncate on writes #ADD YOUR CODE HERE)

For `bq_train_data_op`:

- Set the `task_id` to `bq_valid_data_task`
- Use the `bql_valid` SQL query for the task
- Ensure that the task will use Standard SQL, not Legacy SQL

- Assign the DAG `dag` to the task

bq_valid_data_op = BigQueryOperator(destination_dataset_table="{}.{}_valid_data" .format(DESTINATION_DATASET, model.replace(".", "_")), write_disposition="WRITE_TRUNCATE", # specify to truncate on writes #ADD YOUR CODE HERE)

TODO 3: We will use the python callable `set_new_version_name` to set an Airflow Variable, `NEW_VERSION_NAME`. Complete the code (starting around line 231) for the following `PythonOperator` so that

- We use the Python callable `set_new_version_name`
- Set the `task_id` to `python_new_version_name_task`
- Assign the DAG `dag` to the task

python_new_version_name_op = PythonOperator(provide_context=True, #ADD YOUR CODE HERE)

TODO 4: Using the query (`model_check_sql`) defined around line 296, finish defining the task `bq_check_rmse_query_op` to ensure that the model meets our threshold of an RMSE of 10.0 by defining the `pass_value` argument appropriately. Set the `task_id` to `bq_value_check_rmse_task`

bq_check_rmse_query_op = BigQueryValueCheckOperator(tolerence=0, use_legacy_sql=False, #ADD YOUR CODE HERE) **Note:** The threshold is set artifically high in this lab to ensure the pipeline successfully runs to completion. In practice you may want to change this threshold depending on your business needs.

TODO 5: `bq_check_data_op` should have four downstream tasks: `publish_if_failed_check_op`, `python_new_version_name_op`, `bq_train_data_op`, `bq_valid_data_op`. Define these dependencies to finish the Python script. The dependecies start around line 439.

Task 3. Run DAG in Apache Airflow

In this task you will copy your newly completed DAG into a Cloud Storage bucket, which will be automatically synced with your Cloud Composer environment. Afterwards you will check that your DAG was loaded correctly and start a DAG run.

Check that Cloud Composer environment is ready.

- In the Google Clod Console, on the **Navigation menu** (☰), scroll down to the **Big Data** heading and click on **Composer**.

You should see your Composer environment, `demo-environment`. If it is still being setup, then wait until that process is complete before moving on to the next step.

Define Airflow variables

- Return to Cloud Shell, and run the following command:

gcloud composer environments storage data import \ --source vars.json \ --environment demo-environment \ --location $REGION gcloud composer environments run demo-environment \ --location $REGION variables \ -- \ --i /home/airflow/gcs/data/vars.json **Note:** If the second command above fails, it is possible that your Composer environment is still being set up. Wait until the environment is ready and then try again.

The first command uploads the `vars.json` file to local storage in the Composer environment. The second command runs the `airflow variables` command within the Composer environment. Here we create the `NEW_VERSION_NAME` and `CURRENT_VERSION_NAME` variables and set them to initial values.

Copy DAG into Cloud Storage bucket.

- In Cloud Shell, and run the following commands:

DAGS_FOLDER=$(gcloud composer environments describe demo-environment \ --location $REGION --format="get(config.dagGcsPrefix)") gsutil cp ./chicago_taxi_dag.py ${DAGS_FOLDER}/

Click *Check my progress* to verify the objective. Copy DAG into Cloud Storage bucket

Task 4. Exploring your DAG run in the Airflow UI

Accessing 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.

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.

1. In Airflow, click **chicago_taxi_dag** to open the DAG details page. This page includes several representations of the workflow tasks and dependencies.
2. 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; pink = skipped, etc.).
3. 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
4. Once the status for **ml_engine_training_chicago_taxi_trips_task** has changed to `running`, go to **Navigation menu** > **AI Platform** > **Jobs** and confirm the job is running. This job will take 10-15 minutes to complete.
5. Once all the steps are complete in the DAG, each step has a dark green border or pink border (for tasks that were skipped).

Click *Check my progress* to verify the objective. Exploring DAG runs

Congratulations!

You've successfully deployed a continuous training pipeline using Cloud Composer!

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.

- Overview

NOTES:

/home/airflow/gcs/dags/chicago_taxi_dag.py] invalid syntax (chicago_taxi_dag.py, line 440)