

DT- 7 Automate the process to create, deploy & refine Cloud ML Models

Qwiklabs setup

WHAT YOU'LL NEED

To complete this lab, you'll need:

- Access to a standard internet browser (Chrome browser recommended).
- Time. Note the lab's **Completion** time in Qwiklabs, which is an estimate of the time it should take to complete all steps. Plan your schedule so you have time to complete the lab. Once you start the lab, you will not be able to pause and return later (you begin at step 1 every time you start a lab).
- You do NOT need a Google Cloud Platform account or project. An account, project and associated resources are provided to you as part of this lab.

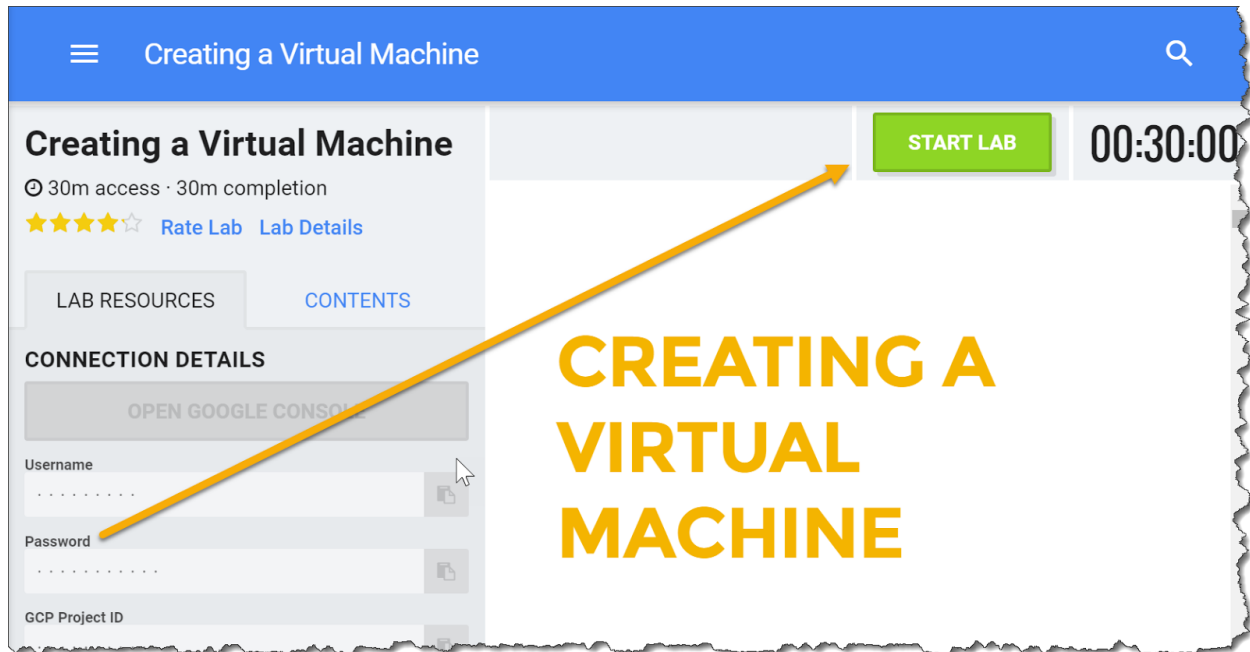
- If you already have your own GCP account, make sure you do not use it for this lab.
- If your lab prompts you to log into the console, **use only the student account provided to you by the lab**. This prevents you from incurring charges for lab activities in your personal GCP account.

Use a new Incognito window (Chrome) or another browser for the Qwiklabs session. Alternatively, you can log out of all other Google / Gmail accounts before beginning the labs.



START YOUR LAB

When you are ready, click **Start Lab**. You can track your lab's progress with the status bar at the top of your screen.



Important: What is happening during this time?

Your lab is spinning up GCP resources for you behind the scenes, including an account, a project, resources within the project, and permission for you to control the resources you will need to run the lab. This means that instead of spending time manually setting up a project and building resources from scratch as part of your lab, you can begin learning more quickly.

FIND YOUR LAB'S GCP USERNAME AND PASSWORD

To access the resources and console for this lab, locate the Connection Details panel in Qwiklabs. Here you will find the account ID and password for the account you will use to log in to the Google Cloud Platform:

CONNECTION DETAILS

OPEN GOOGLE CONSOLE

USERNAME

google822-student@qwiklabs.net

PASSWORD

TZjR4X7B6

If your lab provides other resource identifiers or connection-related information, it will appear on this panel as well.

LOG IN TO GOOGLE CLOUD CONSOLE

Using the Qwiklabs browser tab/window (preferably in Incognito mode) or the separate browser you are using for the Qwiklabs session, copy the Username from the Connection Details panel and click the orange "Open Google Console" button. Paste in the Username, and then the Password as prompted:



Sign in

to continue to Google Cloud Platform

Enter your email

gcpstaging277-student@qwiklabs.net

[More options](#)

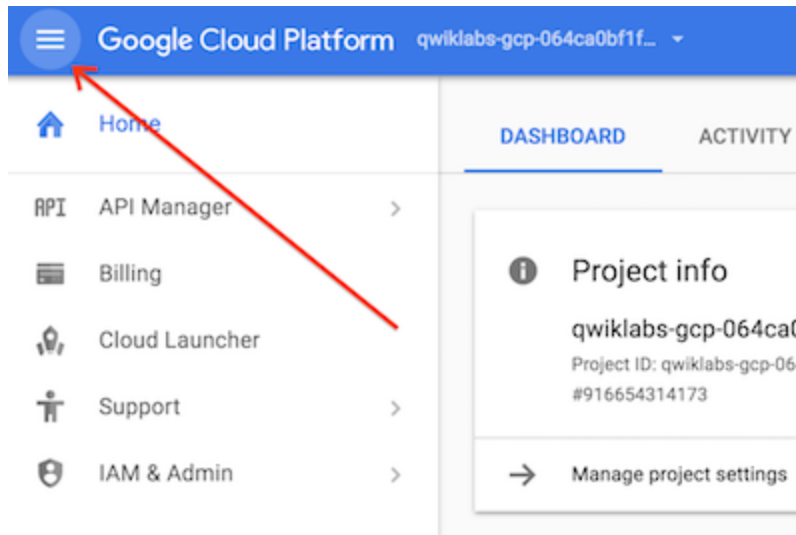
NEXT

Accept the terms and conditions.

Since this is a temporary account, which you will only have access to for this one lab:

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

Note: You can view the menu with a list of GCP Products and Services by clicking the button at the top-left next to "Google Cloud Platform".



Overview

Duration is 1 min

This lab shows you how to deploy a production-ready ML recommendation service on GCP, using Cloud Composer (Airflow) to automate a daily model training and update process. The data for the service is drawn from Google Analytics, via BiqQuery.

By the end of this cloud lab, you will be able to deploy the recommendation API service with GAE, and automate the model update with Cloud Composer (Airflow).

What you'll learn

- How to use Cloud Composer to automate deployment tasks for a production service on GCP
- How to deploy a Recommendation API using Cloud Endpoints
- How to deploy a managed Airflow service running on Kubernetes backed by Cloud SQL

Prerequisites

- Google Cloud Platform Account and a Project with Billing
- Basic Linux Experience

Launch Google Cloud Shell

ACTIVATE GOOGLE CLOUD SHELL

From the GCP Console click the Cloud Shell icon on the top right toolbar:



Then click "Start Cloud Shell":

Google Cloud Shell

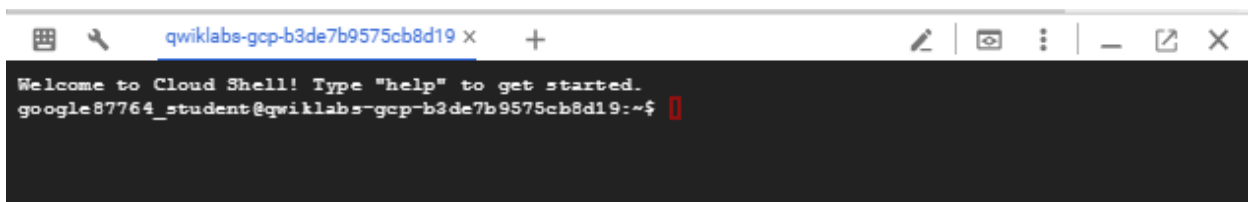
Free, pre-installed with the tools you need for the Google Cloud Platform. [Learn More](#)

```
example-vm-2    europe-west1-b  f1-micro          10.240.119.112  104.155.36.122  RUNN
example-vm-3    us-central1-f    f1-micro          10.240.57.1    104.154.76.241  RUNN
google77703_student@cloudshell:~$
google77703_student@cloudshell:~$ git clone https://github.com/GoogleCloud/appengine-examp
Cloning into 'appengine-example'...
remote: Counting objects: 476, done.
remote: Total 476 (delta 0), reused 0 (delta 0), pack-reused 476
Receiving objects: 100% (476/476), 432.65 KiB | 0 bytes/s, done.
Checking connectivity... done.
google77703_student@cloudshell:~$ cd appengine-example
google77703_student@cloudshell:~/appengine-example$
```

Real Linux environment	Configured for Google Cloud	Popular language support
<ul style="list-style-type: none">• Linux Debian-based OS• 5GB persisted home directory• Add, edit and save files	<ul style="list-style-type: none">• Google Cloud SDK• Google App Engine SDK• Docker• Git• Text editors• Build tools• View more ↗	<ul style="list-style-type: none">• Python• Java• Go• Node.js

[CANCEL](#) [START CLOUD SHELL](#)

It should only take a few moments to provision and connect to the environment:



This virtual machine is loaded with all the development tools you'll need. It offers a persistent 5GB home directory, and runs on the Google Cloud, greatly enhancing network performance and authentication. Much, if not all, of your work in this lab can be done with simply a browser or your Google Chromebook.

Once connected to the cloud shell, you should see that you are already authenticated and that the project is already set to your *PROJECT_ID*:

```
gcloud auth list
```

Command output

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

Note: gcloud is the powerful and unified command-line tool for Google Cloud Platform. Full documentation is available from <https://cloud.google.com/sdk/gcloud>. It comes pre-installed on CloudShell. You will notice its support for tab-completion.

```
gcloud config list project
```

Command output

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

If it is not, you can set it with this command:

```
gcloud config set project <PROJECT_ID>
```

Command output

```
Updated property [core/project].
```

Download the project code and install packages

Duration is 5 min

This codelabs uses the prepared application codes and training data for machine learning. First, you will install ML packages (Miniconda 2 and TensorFlow) on the Cloud Shell after cloning the prepared codes.

Download the lab code

Download and unzip the lab code.

```
cd $HOME
gsutil cp gs://dt7-code/dt7.tgz .
tar xzvf dt7.tgz
```

Install ML packages

Install Miniconda 2.

```
wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
bash Miniconda2-latest-Linux-x86_64.sh
```

Press ENTER to choose the default location for the install. When asked if you wish to make updates to your .bashrc file, say "yes." You do not, however, need to open a new shell as instructed.

```
Do you accept the license terms? [yes|no]
[no] >>> yes
Miniconda2 will now be installed into this location:
/home/gcpstaging10084_student/miniconda2
- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below
[/home/gcpstaging10084_student/miniconda2] >>>
```

```
...
installing: pycparser-2.18-py27hefa08c5_1 ...installing:
pysocks-1.6.7-py27he2db6d2_1 ...installing:
ruamel_yaml-0.11.14-py27h672d447_2 ...installing:
six-1.11.0-py27h5f960f1_1 ...installing: cffi-1.11.2-py27ha7929c6_0
...installing: setuptools-36.5.0-py27h68b189e_0 ...installing:
cryptography-2.1.4-py27h6697b16_0 ...installing:
wheel-0.30.0-py27h2bc6bb2_1 ...installing: pip-9.0.1-py27ha730c48_4
...installing: pyopenssl-17.5.0-py27hcee3be0_0 ...installing:
urllib3-1.22-py27ha55213b_0 ...installing:
requests-2.18.4-py27hc5b0589_1 ...installing: conda-4.3.31-py27_0
...installation finished. Do you wish the installer to prepend the
Miniconda2 install location to PATH in your
/home/gcpstaging10084_student/.bashrc ? [yes|no]
[no] >>> yes
```

Add miniconda to your shell path.

```
export PATH=$HOME/miniconda2/bin:$PATH
```

Create a new miniconda environment, and activate it.

```
cd rec_serve
conda create -y -n recserve
source activate recserve
```

Install conda packages.

```
conda install -y -n recserve --file conda.txt
```

Install additional packages and TensorFlow.

```
pip install tensorflow
pip install -r requirements.txt
```

Upload sample Google Analytics data to BigQuery

Duration is 5 min

Create a Google Cloud Storage bucket.

```
export PROJECT=$(gcloud config get-value project 2> /dev/null)
export BUCKET=gs://recserve_${PROJECT}
gsutil mb ${BUCKET}
```

Copy the sample data to your bucket

```
gsutil -m cp data/ga_sessions_sample.json.gz \
  ${BUCKET}/data/ga_sessions_sample.json.gz
```

Make a new BigQuery dataset. Enter your Qwiklab project ID when you are asked to set a default project.

```
bq mk GA360_test
```

```
Welcome to BigQuery! This script will walk you through the
process of initializing your .bigqueryrc configuration file.
First, we need to set up your credentials if they do not
already exist.
Credential creation complete. Now we will select a default project.
List of projects:
#           projectId           friendlyName
--
1  qwiklabs-resources           Qwiklabs Resources
2  qwiklabs-gcp-83b6977b86f2dbda  qwiklabs-gcp-83b6977b86f2dbda
Found multiple projects. Please enter a selection for
which should be the default, or leave blank to not
```

```
set a default.  
Enter a selection (1 - 2): 2
```

Upload the sample data set to BigQuery. The upload will take around five minutes.

```
bq load --source_format=NEWLINE_DELIMITED_JSON \  
  GA360_test.ga_sessions_sample \  
  ${BUCKET}/data/ga_sessions_sample.json.gz \  
  data/ga_sessions_sample_schema.json
```

Export model training data from BigQuery

Duration is 10 min

The model code takes a CSV file as input, with a header row containing three columns:

- clientId
- contentId
- timeOnPage

To create a CSV training data file, [run](#) the following SQL query to extract those columns from the page tracking event table in BigQuery, and [save the output](#) of the query to a table:

1. Go to the BigQuery web UI.

[GO TO THE BIGQUERY WEB UI](#)

1. Click the **Compose query** button.

2. Enter the SQL query below in the **New Query** text area.
3. Click the **Show Options** button.
4. Click the **Select Table** button in the **Destination Table** section.
5. Select the same dataset that you imported your data to.
6. Enter the table ID "**recommendation_events**", and click **OK**.
7. Click the **Run query** button.

```
SELECT
  fullVisitorId as clientId,
  ArticleID as contentId,
  (nextTime - hits.time) as timeOnPage,
FROM(
  SELECT
    fullVisitorId,
    hits.time,
    MAX(IF(hits.customDimensions.index=10,
hits.customDimensions.value,NULL)) WITHIN hits AS ArticleID,
    LEAD(hits.time, 1) OVER (PARTITION BY fullVisitorId, visitNumber
ORDER BY hits.time ASC) as nextTime
  FROM [GA360_test.ga_sessions_sample]
  WHERE hits.type = "PAGE"
) HAVING timeOnPage is not null and contentId is not null;
```

Note that the article ID in the sample data set is stored in a custom dimension with index of 10. This is used in the inner select clause to filter the events. Also note that "time on page" is calculated by subtracting the next event for the user ID fullVisitorId from the page hit event.

[Export the destination table](#) to a CSV file in the Cloud Storage bucket you created:

1. Go to the BigQuery web UI.

[GO TO THE BIGQUERY WEB UI](#)


1. In the navigation, find the GA_360_test dataset to which you uploaded the sample data, and then expand to display its contents.

▼ GA360_test



recommendation_events



1. Click the down arrow icon  next to the destination table from the previous step.
2. Select **Export table** to display the **Export to Google Cloud Storage** dialog.
3. Leave the default settings in place for **Export format** and **Compression** (**CSV** and **None**, respectively).
4. In the **Google Cloud Storage URI** textbox, enter a URI in the format `gs://[your_bucket]/data/recommendation_events.csv`, where `[your_bucket]` is the Cloud Storage bucket you created earlier.
5. Click **OK** to export the table. While the job is running, **(extracting)** appears next to the name of the table in the navigation.

To check on the progress of the job, look near the top of the navigation for **Job History** for an **Extract** job.

Now download the CSV file you just exported from BigQuery:

```
gsutil cp ${BUCKET}/data/recommendation_events.csv \
data/recommendation_events.csv
```

Take a look at the data. It consists of the three columns: `clientId`, `contentId` (i.e. article ID) and `timeOnPage` in milliseconds. Each row corresponds to a page view event by a particular user.

```
head data/recommendation_events.csv
```

```
clientId,contentId,timeOnPage
1049615920436892829,299957318,96370
1397611698544709263,299933565,46702
1414839705705236630,299965853,35263
1501431610796983806,299836255,673917
```

```
2066465362125086010,299933565,78308  
2576389842575578331,299772450,89613  
2676126580987099715,299865757,454411  
2955268363965511895,299918253,147807  
3589291144131971323,299907275,64207
```

Create an initial WALS recommendation model

Duration is 10 min

In the production system, Airflow will train a model nightly using the latest data in BigQuery, exported from Google Analytics.

For the initial deployment, however, you manually train the model and upload the resulting model files to your bucket.

Run the following to train the model.

```
cd wals_ml_engine  
./mltrain.sh local ../data/recommendation_events.csv \  
--data-type web_views --use-optimized
```

This will take a couple minutes, and create a job directory under `wals_ml_engine/jobs` like `"wals_ml_local_20180102_012345/model"`, containing the model files saved as numpy arrays.

Copy the model files from this directory to the model folder in the project bucket:


```
export JOB_MODEL=$(find jobs -name "model" | tail -1)
gsutil cp ${JOB_MODEL}/* ${BUCKET}/model/
```

The model training code needs to live in a folder on GCS, so that Airflow can run the training task. Generate a python distributable package, and copy that to a code folder in your bucket.

```
python setup.py sdist
gsutil cp dist/wals_ml_engine-0.1.tar.gz ${BUCKET}/code/
cd ..
```

Deploy the recommendation API service

Duration is 10 min

Deploy the recommendation API service on GAE Flexible environment.

```
gcloud app create --region=us-east1
gcloud app update --no-split-health-checks
cd scripts
./deploy_api.sh      # Deploy the API.
./deploy_app.sh      # Deploy the backend app.
```

Once the service is successfully deployed, you can test it with the following script.

```
./query_api.sh
cd ..
```

You will see the result as below.

```
curl
"https://[PROJECT_ID].appspot.com/recommendation?userId=54485436471
76335931&numRecs=5"
```

```
{  
  "articles": [  
    "299941605",  
    "299800704",  
    "298299208",  
    "299800661",  
    "299928862"  
  ]  
}
```

Note

- Since the deployment process takes around 10 minutes, you can open a new SSH console (Cloud Shell terminal) and proceed with the next step (Airflow installation) in parallel.

Deploy the managed Airflow service

Duration is 10 min

If you opened a new SSH console (Cloud Shell terminal) for this step, make sure to activate the recserve environment and change the current directory to rec_serve.

```
cd $HOME  
export PATH=$HOME/miniconda2/bin:$PATH  
source activate recserve  
cd rec_serve
```

Deploy the managed Airflow service using Google Kubernetes Engine.

```
cd airflow/deploy
./deploy_airflow.sh
cd ../../
```

Create "dags", "logs" and "plugins" folders in the GCS bucket created by the deploy script named managed-airflow-{random hex value}, e.g.

gs://managed-airflow-e0c99374808c4d4e8002e481. See this [screenshot](#). The name of the bucket is available in the ID field of the

airflow/deploy/deployment-settings.yaml file created by the deploy script. You can create the folders in the cloud console, or use the following script:

```
python airflow/deploy/create_buckets.py
```

Copy training.py to the dags folder in your airflow bucket.

```
export AIRFLOW_BUCKET=`python -c "\
import yaml;\
f = open('airflow/deploy/deployment-settings.yaml');\
settings=yaml.load(f);\
f.close();\
print settings['id']" `
```

```
gsutil cp airflow/dags/training.py gs://${AIRFLOW_BUCKET}/dags
```

Copy plugins to the plugins folder of your airflow bucket.

```
gsutil cp -r airflow/plugins gs://${AIRFLOW_BUCKET}
```

Restart the airflow webserver pod. This is necessary when you have updated the plugins in an already running Airflow cluster.

```
WS_POD=`kubectl get pod | grep -o airflow-webserver-[0-9]*-[0-9a-z]*`
kubectl get pod ${WS_POD} -o yaml | kubectl replace --force -f -
```

Using Airflow

Duration is 10 min


You can find the URL and login credentials for the airflow admin interface in the file

`airflow/deploy/deployment-settings.yaml`.

```
cat airflow/deploy/deployment-settings.yaml
```

```
...  
web_ui_password: IiDYrpwJcT...  
web_ui_url: http://35.226.101.220:8080  
web_ui_username: airflow
```

Navigate to the Airflow admin app and login with your credentials. You will see the Airflow console home page:

 Airflow

DAGs

Data Profiling ▾

Browse ▾

Admin ▾

Docs ▾

About ▾

DAGs

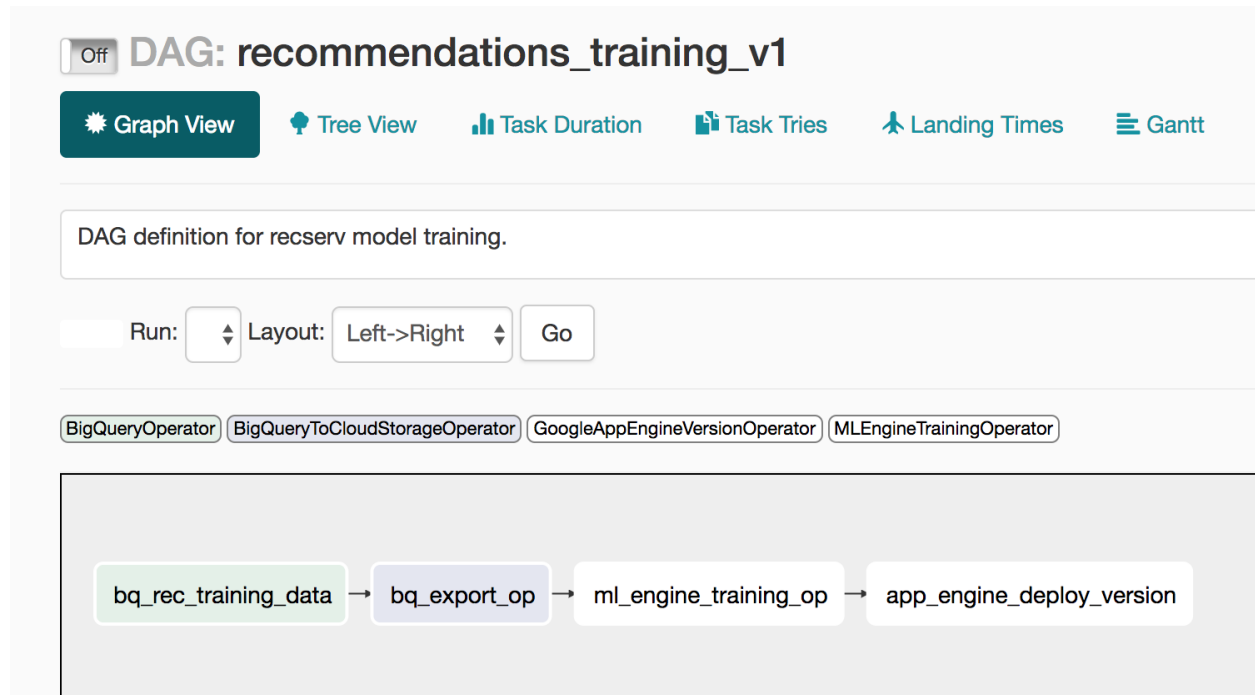
Show entries

		DAG	Schedule	Owner	Recent Task
	<input type="checkbox"/> Off	recommendations_training_v1	00 21 ***	airflow	  

Showing 1 to 1 of 1 entries

[Hide Paused DAGs](#)


Click on the link for our DAG, recommendations_training_v1. Click on the "Graph View" to see the graphical view of each task in the DAG and the dependencies between them:



The tasks are:

1. **bq_rec_training_data**: Run a BQ query against the dataset containing the tables imported from GA360. Dump that data into a table "recommendation_events".
2. **bq_export_op**: Export the data into a CSV file in GCS.
3. **ml_engine_training_op**: Train an ML model using the CSV file of event data as input. The training task writes new model files to the GCS model directory that is read by the App Engine endpoint.
4. **app_engine_deploy_version**: Redeploy the App Engine Endpoint app, so that the new model files are loaded. Traffic is migrated to the new app automatically by App Engine.

Now click on the first task of the dag, bq_rec_training_data:

bq_rec_training_data  on 2018-01-10

Task Instance Details **Rendered** **Task Instances** **View Log**

Run Ignore All Deps Ignore Task State Ignore Task Deps

Clear Past Future Upstream **Downstream** Recursive

Mark Success Past Future Upstream Downstream

Run that task by clicking on the "Run" button.

You can run the other tasks in the graph in similar fashion. Explore the rest of the Airflow interface.

Conclusion

Duration is 1 min

This lab covered all the steps necessary to deploy a production ready recommendation service on Google Cloud Platform.

What we've covered

- How to install a miniconda environment in Cloud Shell
- How to upload data to BigQuery from Cloud Shell
- Training the WALS model on data exported from BigQuery
- Deploying a Google App Engine endpoint server to serve recommendations
- Installing an Airflow service on Kubernetes
- Inspecting and running Airflow DAG tasks from the Airflow console

Wrapping Up

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.