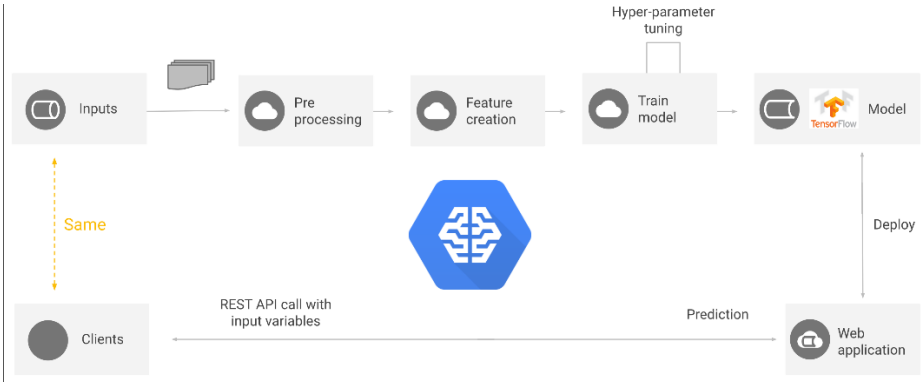
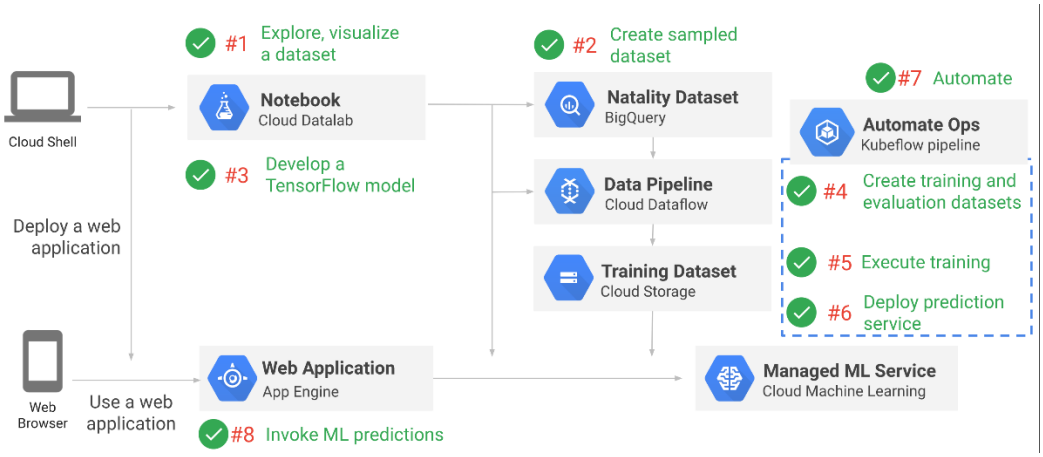
Machine learning class – Vinicius F. Caridá

End-to-End machine learning solution





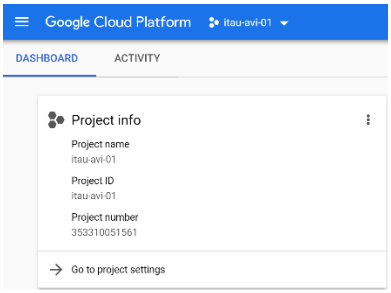
I recommend read and do the TensorFlow Get Started examples

https://www.tensorflow.org/tutorials?hl=pt_BR

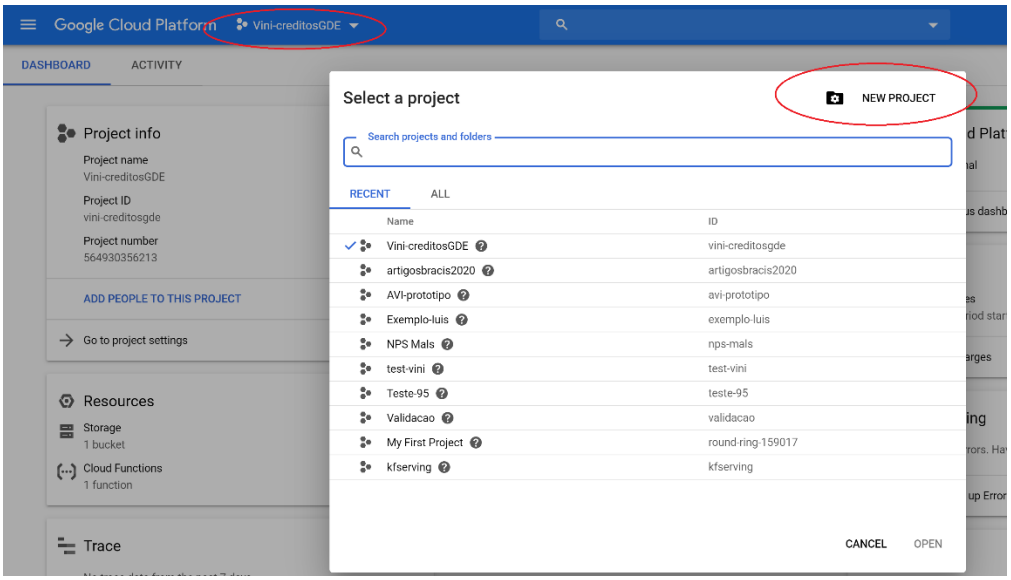
0 – INICIAL CONFIGURATION (15 min)

Go to gcp console https://console.cloud.google.com

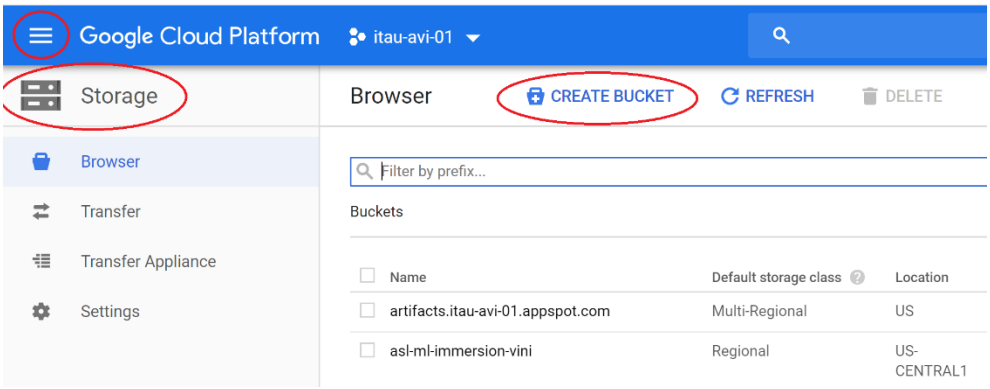
From the GCP First access: Hamburger menu -> home and get the informations “Project name” and “Project ID”



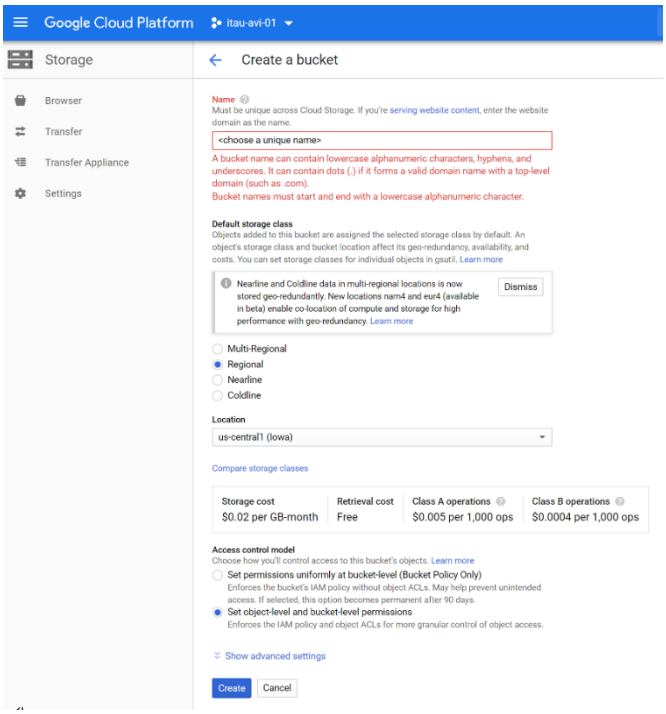
If you don’t have a project yet, please, create one accessing:



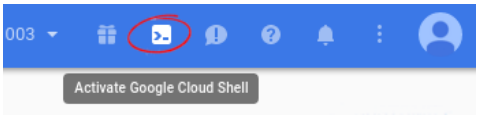
Then access: hamburger menu -> Storage and click on “create bucket”



Choose a unique name to your bucket, select the “regional” option and choose one location. I selected the “US-CENTRAL1”



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

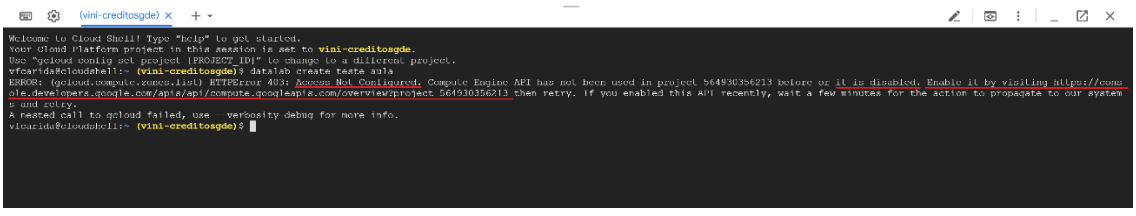


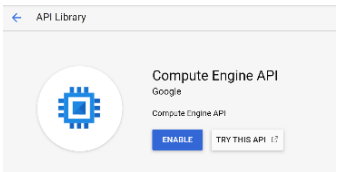
Then click "Start Cloud Shell":

In Cloud Shell, type:

```
datalab create datalab-instance-name (escolha um nome)
```

If you get a error “access not configured”, like bellow, just click in the link and enable compute engine API.





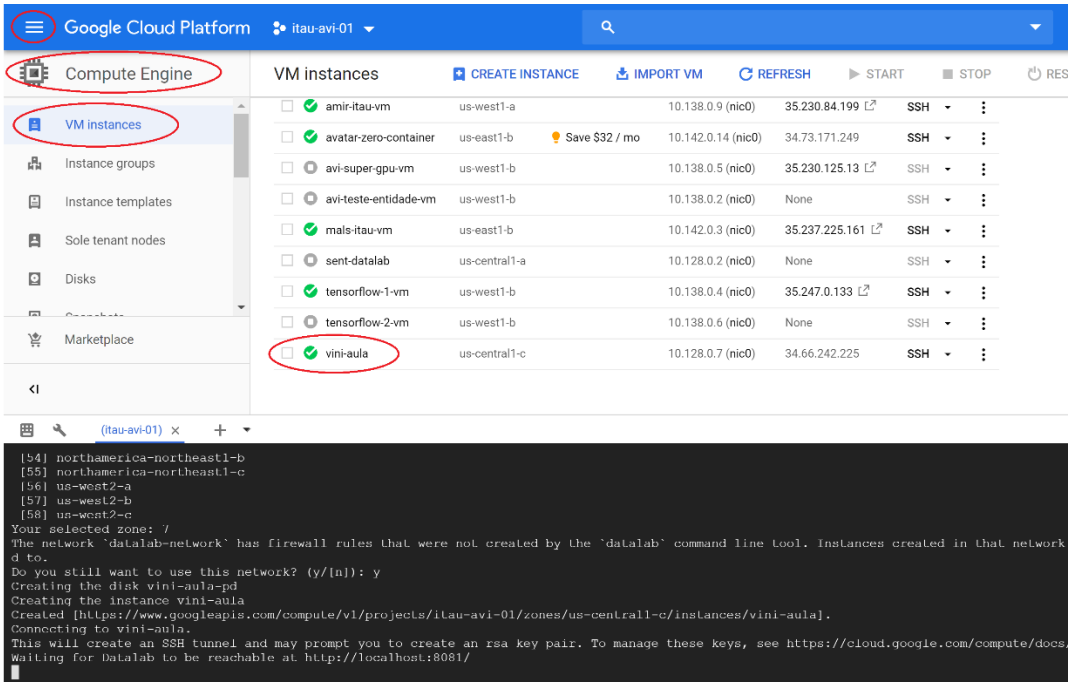
PS: Simple error like this is common in the first project, because you don't have de APIs enabled yet. Stay tuned!

Select zone <I selected the zone 7 because is the same zone that my bucket>

PS: I recommend don't register password (make it simple after)

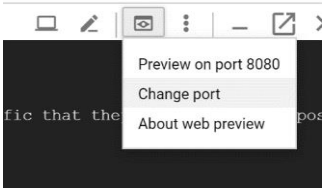
When the process are complete, access:

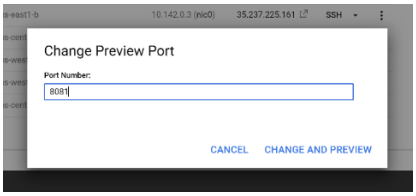
Hamburger menu -> Compute Engine -> VM instances and chek if your machine are ready



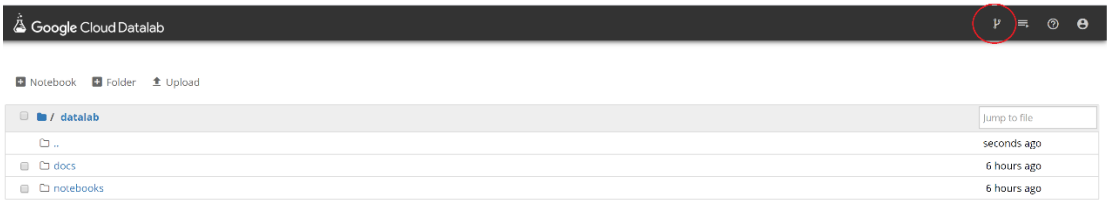
PS: If in any moment you lost the connection with the server (you closed the cloud shell, close the browser, etc.), open cloud shell again and use the command: datalab connect ***datalab-instance-name***

Open the: web preview -> change port -> 8081 -> change and preview





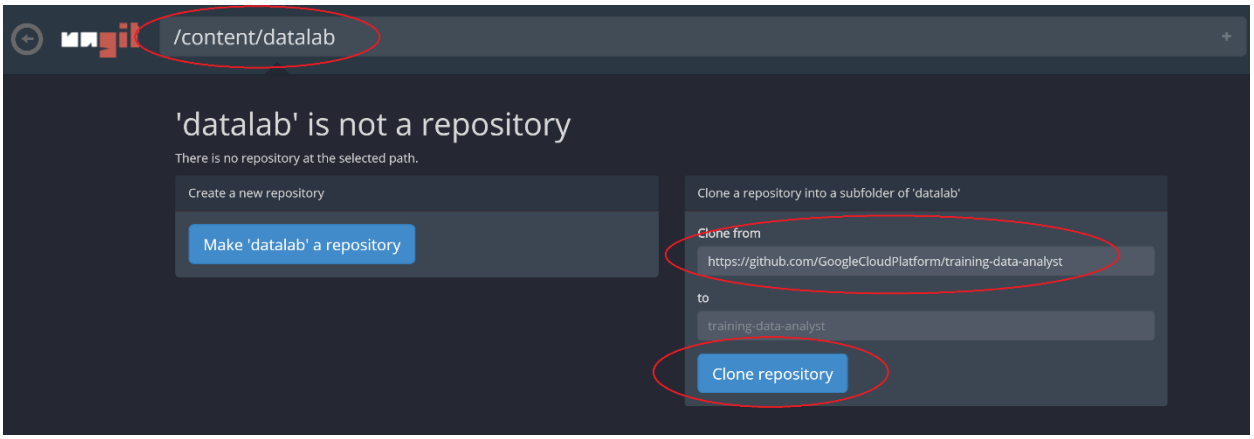
Click on ungit bottom



Insert the path of repository: /content/datalab

Insert the url of the repository: https://github.com/GoogleCloudPlatform/training-data-analyst

Click in “clone repository”

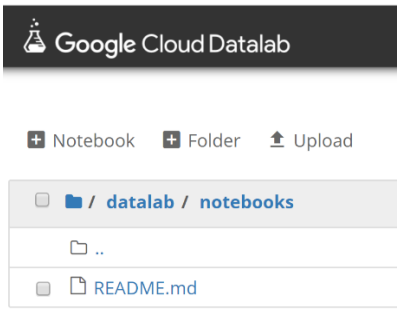


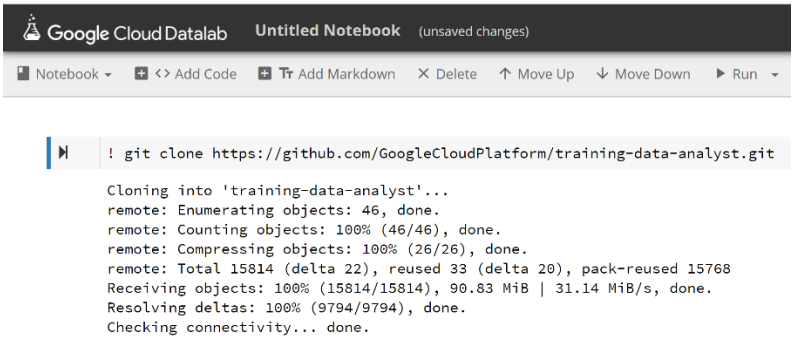
You don’t need to do this party. It is only another way to clone the repository

Navigate to: datalab -> Click on: +Notebook

Insert and run the line below to clone a repository

! git clone https://github.com/GoogleCloudPlatform/training-data-analyst.git





Navigate to: datalab -> notebooks -> Click on: +Folder

In this new folder, do the upload of all files of zip file “tensor_keras_pytorch.zip”

---

0 – RUN ALL SOLUTION (45 min)

In Cloud Datalab, navigate to **notebooks/training-data-analyst/courses/machine_learning/deepdive/06_structured**

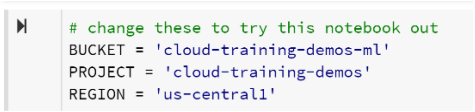
In this folder we have all previous exercises with the answers. Now, go to each one python notebook (1 to 5 bellow), read very quickly the narrative and execute each cell in turn.

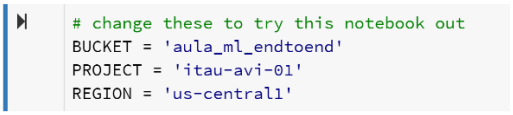
In the first moment, the objective here is not understand the code. Please, read very quickly each cell and rull all.

Open new table on cloud shell

In Cloud Shell (**NOT Datalab**), run the code: gsutil -m cp -r gs://cloud-training-demos/babyweight/preproc gs://**your-bucket/**

Collect the data of your project. We will need change this information in all python notebooks





1 – EXPLORER (20 min)

In Cloud Datalab, navigate to **/training-data-analyst/courses/machine_learning/deepdive/06_structured/labs** and open **1_explore.ipynb**.

PS: If you go into the labs folder, you need to complete some exercises. In “06_structured” folder, the code is complete.

analyst/courses/machine_learning/deepdive/06_structured/labs
analyst/courses/machine_learning/deepdive/06_structured/

In Datalab, change the variables in the first code cell for yours informations

```
# change these to try this notebook out
BUCKET = 'cloud-training-demos-ml'
PROJECT = 'cloud-training-demos'
REGION = 'us-central1'
```

```
# change these to try this notebook out
BUCKET = 'aula_ml_endtoend'
PROJECT = 'itau-avi-01'
REGION = 'us-central1'
```

Click on **Clear -> All Cells**. Now read the narrative and execute each cell in turn.

More details to natality dataset, looking:
https://bigquery.cloud.google.com/table/publicdata:samples.natality

Home work: go to the labs folder and try to do lab task #1, lab task #2 and other analysis/plots to explore and understand the dataset. Write a short conclusion of your data analysis

2 – SAMPLE (20 min.)

In Cloud Datalab, click on the Home icon, and then navigate to **notebooks/training-data-analyst/courses/machine_learning/deepdive/06_structured/** and open **2_sample.ipynb**.

Again, change the variables in the first code cell for yours informations. Click on **Clear -> All Cells**. Now read the narrative and execute each cell in turn.

Home work: go to the labs folder and try to do lab task #1, lab task #2, lab task #3 and other preprocessing in the dataset. Write a short conclusion of your view of the preprocessing

3 – TENSORFLOW (30 min.)

In Cloud Datalab, click on the Home icon, and then navigate to **notebooks/training-data-analyst/courses/machine_learning/deepdive/06_structured/labs** and open **3_tensorflow.ipynb**.

Again, change the variables in the first code cell for yours informations. Click on **Clear -> All Cells**. Now read the narrative and execute each cell in turn.

Home work: go to the labs folder and try to do lab task #1, lab task #2, lab task #3, lab task #4, lab task #5

More information of tensorflow:

https://www.youtube.com/watch?v=er8RQZoX3yk&list=PL9Hr9sNUjfsmEu1ZniY0XpHSzl5uihcXZ

https://github.com/Hvass-Labs/TensorFlow-Tutorials

paper: https://arxiv.org/abs/1708.02637

4 – PRE-PROCESSING (15 min.)

In Cloud Datalab, click on the Home icon, and then navigate to **notebooks/training-data-analyst/courses/machine_learning/deepdive/06_structured/** and open **4_preproc.ipynb**.

Again, change the variables in the code cell for yours informations. Click on **Clear -> All Cells**. Now read the narrative and execute each cell in turn.

Home work: go to the labs folder and try to do the ToDos

```
If you get the error: module 'six' has no attribute 'ensure_str'
Install: !pip install six==1.12.0
Restart the notebook
```

---

5 – TRAINING (30 min.)

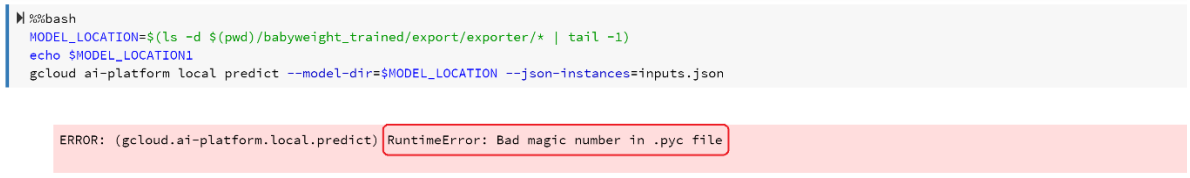
In Cloud Datalab, click on the Home icon, and then navigate to **notebooks/training-data-analyst/courses/machine_learning/deepdive/06_structured/** and open **5_train.ipynb**.

Again, change the variables in the code cell for yours informations. Click on **Clear -> All Cells**. Now read the narrative and execute each cell in turn.

Please, in the first time, please, run only until Hyperparameter tuning. Don't run nothing after the title "Hyperparameter tuning"

Home work: go to the labs folder and try to do lab task #1, lab task #2, lab task #3, lab task #4, lab task #5

If you get the error: RuntimeError: Bad magic number in .pyc file



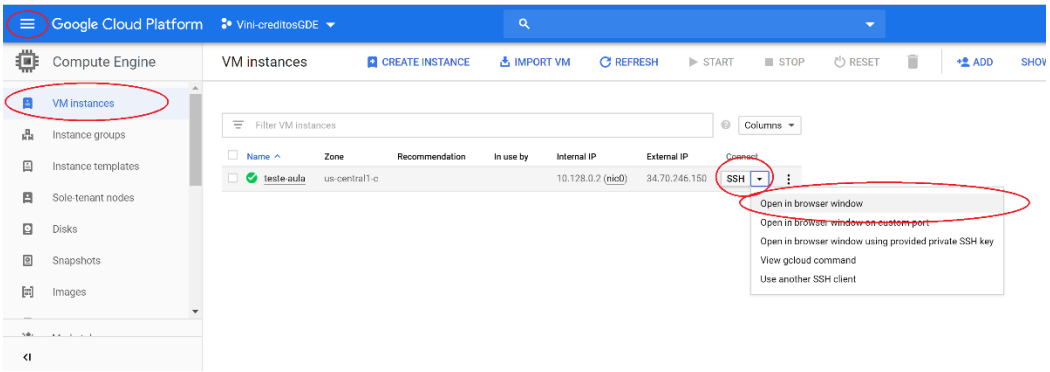
Insert the code “--verbosity debug” like bellow
Copy the directory address

```
%%bash
MODEL_LOCATION=$(ls -d $(pwd)/babyweight_trained/export/exporter/* | tail -1)
echo $MODEL_LOCATION1
gcloud ai-platform local predict --model-dir=$MODEL_LOCATION --json-instances=inputs.json --verbosity debug
```

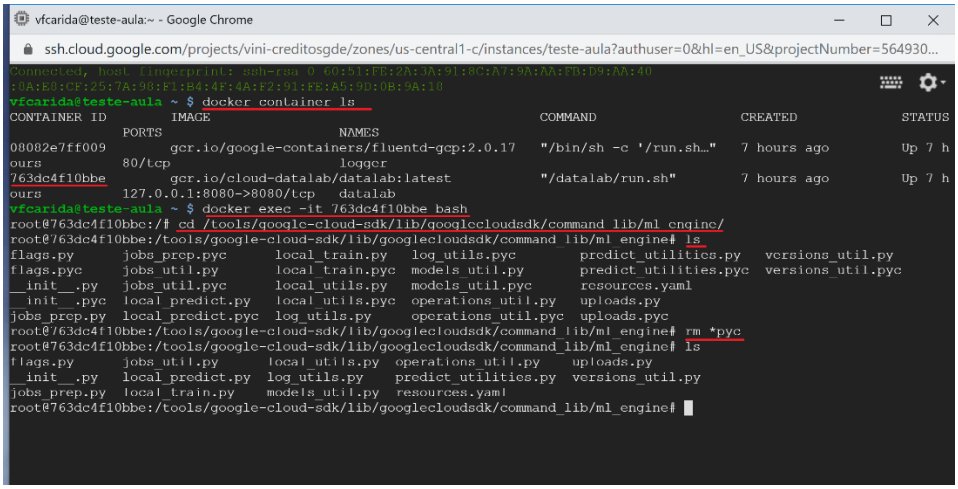
```
DEBUG: Running [gcloud.ai-platform.local.predict] with arguments: [--json-instances: "inputs.json", --model-dir: "/content/datalab/training-data-analys
t/courses/machine_learning/deepdive/06_structured/babyweight_trained/export/exporter/1583101117", --verbosity: "debug"]
DEBUG: (gcloud.ai-platform.local.predict) RuntimeError: Bad magic number in .pyc file
Traceback (most recent call last):
  File "/tools/google-cloud-sdk/lib/googlecloudsdk/calliope/cli.py", line 985, in Execute
    resources = calliope_command.Run(cli=self, args=args)
  File "/tools/google-cloud-sdk/lib/googlecloudsdk/calliope/backend.py", line 795, in Run
    resources = command_instance.Run(args)
  File "/tools/google-cloud-sdk/lib/surface/ai_platform/local/predict.py", line 79, in Run
    signature_name=args.signature_name)
  File "/tools/google-cloud-sdk/lib/googlecloudsdk/command_lib/ml_engine/local_utils.py", line 110, in RunPredict
    raise LocalPredictRuntimeError(err)
LocalPredictRuntimeError: RuntimeError: Bad magic number in .pyc file

ERROR: (gcloud.ai-platform.local.predict) RuntimeError: Bad magic number in .pyc file
```

Go back to console and access: Hamburger menu -> vm Instances -> SSH -> Open in browser window



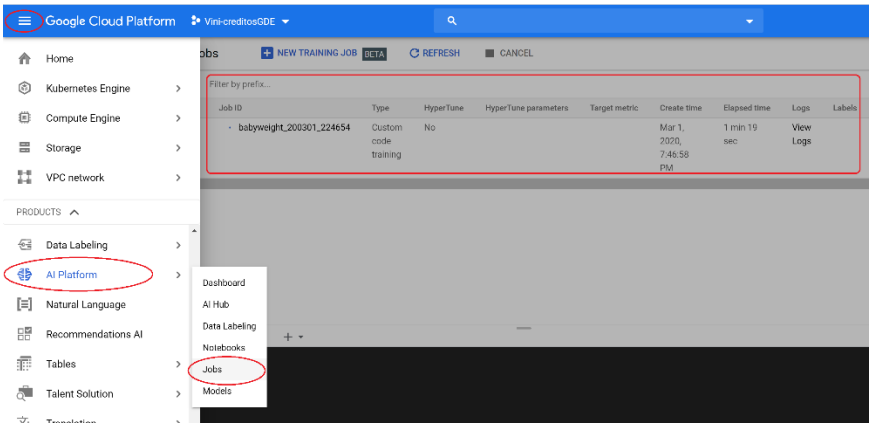
-Use the code to see the container ID: docker container ls
-Access the container: docker exec -it [container-id] bash
-Go to the directory that you copied on jupyter notebook cd /tools/google-cloud-sdk/lib/googlecloudsdk/command_lib/ml_engine/
-Remove all pyc files: rm *.pyc



Close SSH, go back to jupyter notebook and run the cell again

---

access: Hamburger menu -> AI Platform -> Jobs and confirm if the jobs was start



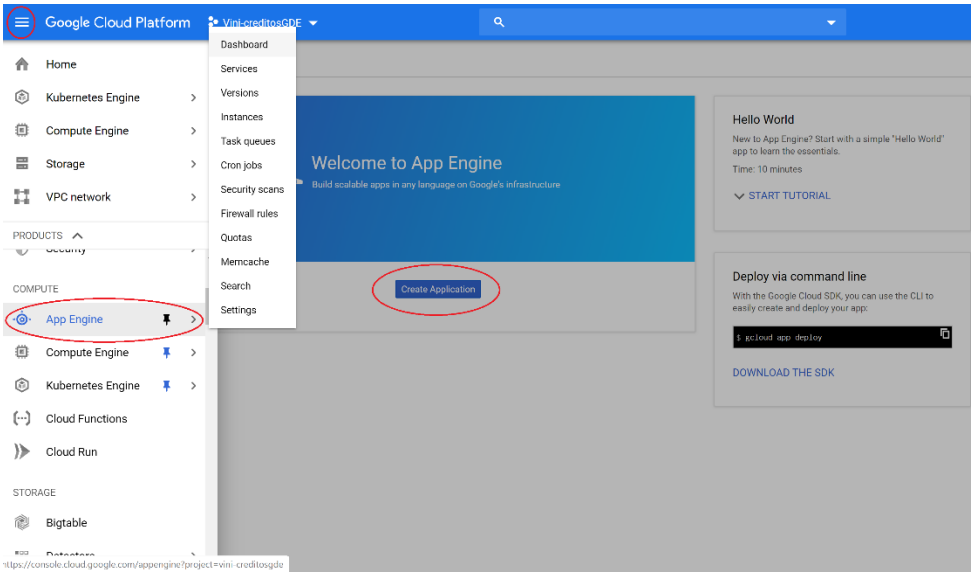
Once the training started, it tak around 2 hours. Please, go back to topic one and now run each cell again more careful.

6 – DEPLOY (20 min.)

In Cloud Datalab, click on the Home icon, and then navigate to **notebooks/training-data-analyst/courses/machine_learning/deepdive/06_structured/labs** and open **6_deploy.ipynb**.

Again, change the variables in the code cell for yours informations. Click on **Clear -> All Cells**. Now read the narrative and execute each cell in turn.

Home work: go to the labs folder and try to do lab task #1, lab task #2, lab task #3, lab task #4



Build an AppEngine app to serve ML predictions

Add new cloud shell session

Run the command line: git clone https://github.com/GoogleCloudPlatform/training-data-analyst.git

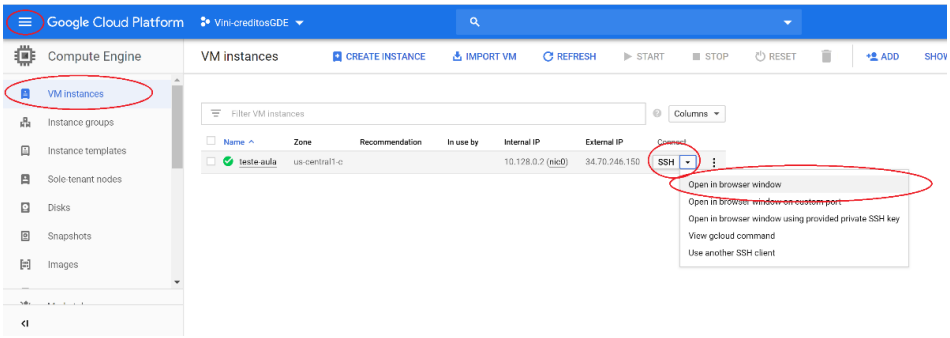
```
vfcarida@cloudshell:~ (vini-creditosgde)$ git clone https://github.com/GoogleCloudPlatform/training-data-analyst.git
Cloning into 'training-data-analyst'...
remote: Enumerating objects: 1234, done.
remote: Counting objects: 100% (1234/1234), done.
remote: Compressing objects: 100% (901/901), done.
remote: Total 32243 (delta 310), reused 1141 (delta 249), pack-reused 31009
Receiving objects: 100% (32243/32243), 328.92 MiB | 31.28 MiB/s, done.
Resolving deltas: 100% (19495/19495), done.
Checking out files: 100% (8446/8446), done.
vfcarida@cloudshell:~ (vini-creditosgde)$ ls
README-cloudshell.txt  training-data-analyst
vfcarida@cloudshell:~ (vini-creditosgde)$
```

In Cloud Shell (**NOT Datalab**), navigate to the folder containing the starter code for this lab

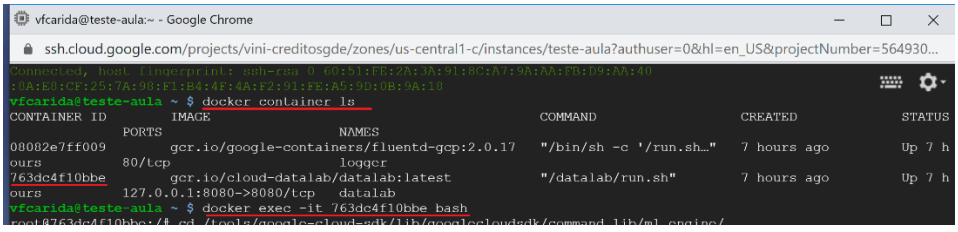
```
cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving
```

You don't need to do this party. It is only a way access the data buy container

Go back to console and access: Hamburger menu -> vm Instances -> SSH -> Open in browser window



-Use the code to see the container ID: docker container ls
-Access the container: docker exec -it [container-id] bash
-Go to the directory: cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving



When model training ends, you can go direct to the topic 6 – deploy model and see the application working.

To see and learning how to build/edit the application, follow each step below.

1. Fix the code

Step 1

Run the what_to_fix.sh script to see a list of items you need to add/modify to existing code to run your app:

```
./what_to_fix.sh
```

As a result of this, you will a list of filenames and lines within those files marked with "TODO". These are the lines where you have to add/modify code. For this lab, you will focus on #*TODO* items for main.py and form.html only.

Step 2

You may use the Cloud Shell code editor to view and edit the contents of these files.

Click on the  icon on the top right of your Cloud Shell window to launch Code Editor

Once launched, navigate to the ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/labs/serving directory.

Step 3

Open the **application/main.py** and **application/templates/form.html** files and replace #*TODO*s with code. For hints, see the following section.

2. Hints to modify main.py

Step 1

Set the credentials to use Google Application Default Credentials and specify the ML Engine API with version.

Step 2

Specify the name of your trained model deployed on Cloud MLE using the **parent** variable.

Step 3

Build the call request with the **prediction** variable.

Step 4

Cast the gestation_weeks feature into a float within the **features** array.

3. Hints to modify form.html

Step 1

Add more values for the drop down option for **Plurality**.

4. Code to modify main.py

Step 1

Open the main.py file by clicking on it. Notice the lines with # *TODO* for setting credentials and the api to use.

Set the credentials to use Google Application Default Credentials (recommended way to authorize calls to our APIs when building apps deployed on AppEngine):

```
credentials = GoogleCredentials.get_application_default()
```

Specify the api name (ML Engine API) and version to use:

```
api = discovery.build('ml', 'v1', credentials=credentials)
```

Step 2

Scroll further down in main.py and look for the next #*TODO* in the method get_prediction(). In there, specify, using the **parent** variable, the name of your trained model deployed on Cloud MLE:

```
parent = 'projects/%s/models/%s' % (project, model_name)
```

Step 3

Now that you have all the pieces for making the call to your model, build the call request by specifying it in the **prediction** variable:

```
prediction = api.projects().predict(body=input_data, name=parent).execute()
```

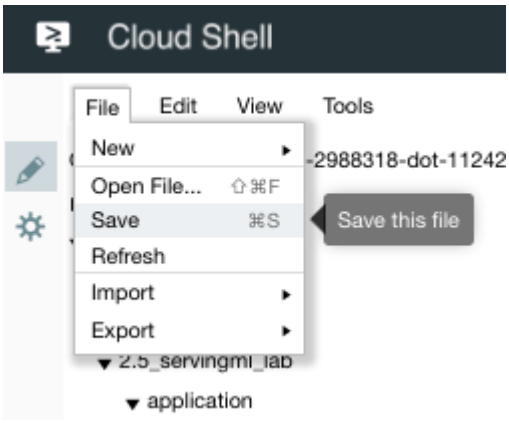
Step 4

The final #*TODO* (scroll towards bottom) is to get gestation_weeks from the form data and cast into a float within the **features** array:

```
features['gestation_weeks'] = float(data['gestation_weeks'])
```

Step 5

Save the changes you made using the **File** > **Save** button on the top left of your code editor window.



5. Code to modify form.html

form.html is the front-end of your app. The user fills in data (features) about the mother based on which we will make the predictions using our trained model.

Step 1

In code editor, navigate to the `application/templates` directory and click to open the `form.html` file

Step 2

There is one #TODO item here. Look for the div segment for **Plurality** and add options for other plurality values (2, 3, etc)

```
<md-option value="2">Twins</md-option>
<md-option value="3">Triplets</md-option>
```

Step 3

Save the changes you made using the **File** > **Save** button on the top left of your code editor window

6. Deploy and test your app

navigate to the /training-data-analyst/courses/machine_learning/deepdive/06_structured/serving/application/

Step 1

In Cloud Shell, go to /training-data-analyst/courses/machine_learning/deepdive/06_structured/serving/

run the deploy.sh script to install required dependencies and deploy your app engine app to the cloud.

```
./deploy.sh
```

Step 2

Go to the url `https://<PROJECT-ID>.appspot.com` and start making predictions.

Note: Replace <PROJECT-ID> with your Project ID.

Extra:

`Serving ML Predictions in batch and real-time`

Step 1

In Cloud Shell, navigate to the folder containing the starter code for this lab

```
cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/labs/serving
```

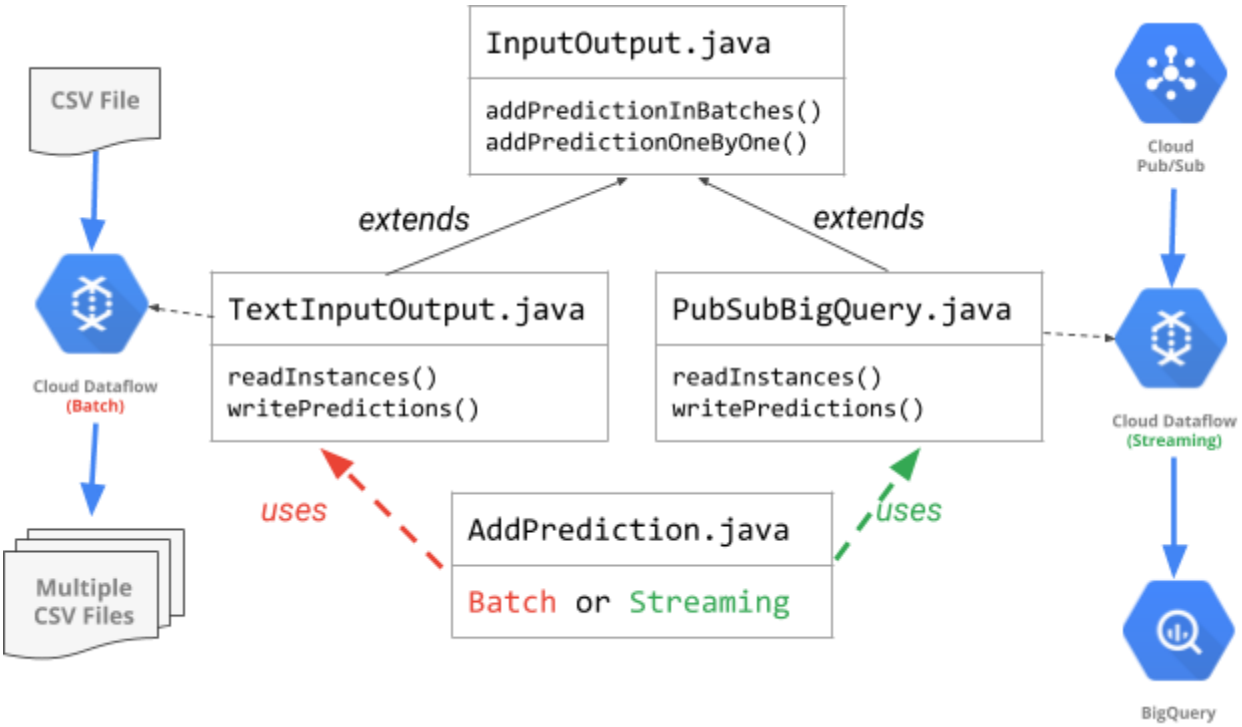
Step 2

Run the what_to_fix.sh script to see a list of items you need to add/modify to existing code to run your app:

```
./what_to_fix.sh
```

As a result of this, you will see a list of filenames and lines within those files marked with "TODO". These are the lines where you have to add/modify code. For this lab, you will focus on #TODO items for **.java files only**, namely BabyweightMLService.java : which is your prediction service

7. How the code is organized



8. Prediction service

In this section, you fix the code in **BabyweightMLService.java** and test it with the **run_once.sh** script that is provided. If you need help with the code, look at the next section that provides hints on how to fix code in BabyweightMLService.java

Step 1

You may use the Cloud Shell code editor to view and edit the contents of these files.

Click on the  icon on the top right of your Cloud Shell window to launch Code Editor

Step 2

After it is launched, navigate to the following directory: training-data-analyst/courses/machine_learning/deepdive/06_structured/labs/serving/pipeline/src/main/java/com/google/cloud/training/mlongcp

Step 3

Open the **BabyweightMLService.java** files and replace #*TODO*s in the code.

Step 4

Once completed, go into your Cloud Shell and run the run_once.sh script to test your ML service

```
cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving
./run_once.sh
```

9. Serve predictions for batch requests

This section of the lab calls AddPrediction.java that takes a batch input (one big CSV), calls the prediction service to generate baby weight predictions and writes them into local files (multiple CSVs).

Step 1

In your Cloud Shell code editor, open the **AddPrediction.java** file available in the following directory: training-data-analyst/courses/machine_learning/deepdive/06_structured/labs/serving/pipeline/src/main/java/com/google/cloud/training/mlongcp

Step 2

Look through the code and notice how, based on input argument, it decides to set up a batch or streaming pipeline, and creates the appropriate TextInputOutput or PubSubBigQuery io object respectively to handle the reading and writing.

Note: Look back at the diagram in "how code is organized" section to make sense of it all.

Step 3

Test batch mode by running the run_ontext.sh script provided in the lab directory:

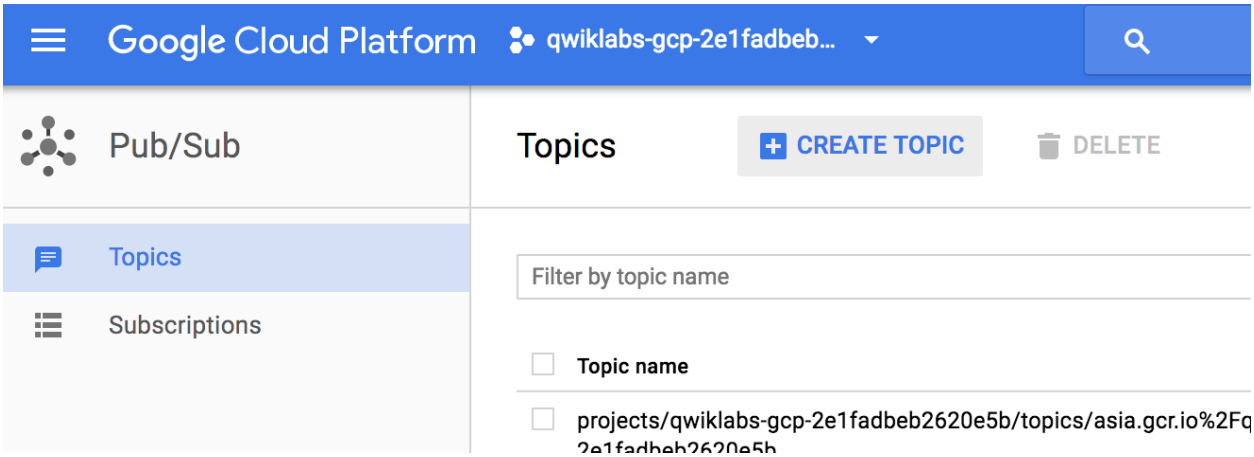
```
cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving
./run_ontext.sh
```

10. Serve predictions real-time with a streaming pipeline

In this section of the lab, you will launch a streaming pipeline with Dataflow, which will accept incoming information from Cloud Pub/Sub, use the info to call the prediction service to get baby weight predictions, and finally write that info into a BigQuery table.

Step 1

On your GCP Console's left-side menu, go into Pub/Sub and click the "Create Topic" button on top. Create a topic called **babies**.



Step 2

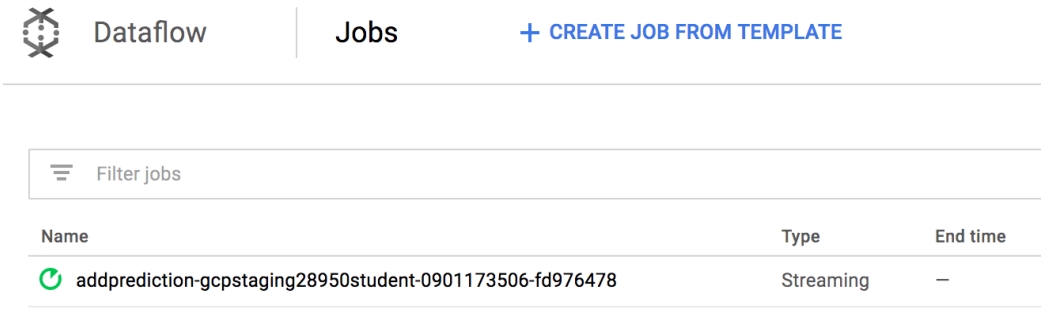
Back in your Cloud Shell, modify the script **run_dataflow.sh** to get Project Id (using *--project*) from command line arguments, and then run as follows:

```
cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving
./run_dataflow.sh
```

This will create a streaming Dataflow pipeline.

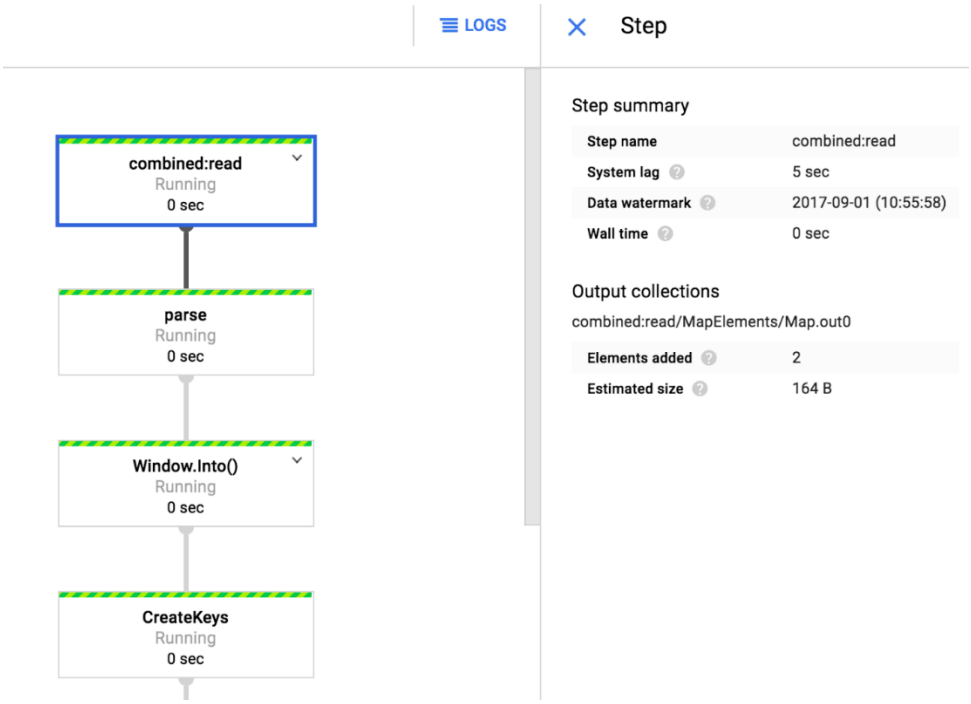
Step 3

Back in your GCP Console, use the left-side menu to go into Dataflow and verify that the streaming job is created.



Step 4

Next, click on the job name to view the pipeline graph. Click on the pipeline steps (boxes) and look at the run details (like system lag, elements added, etc.) of that step on the right side.



This means that your pipeline is running and waiting for input. Let's provide input through the Pub/Sub topic.

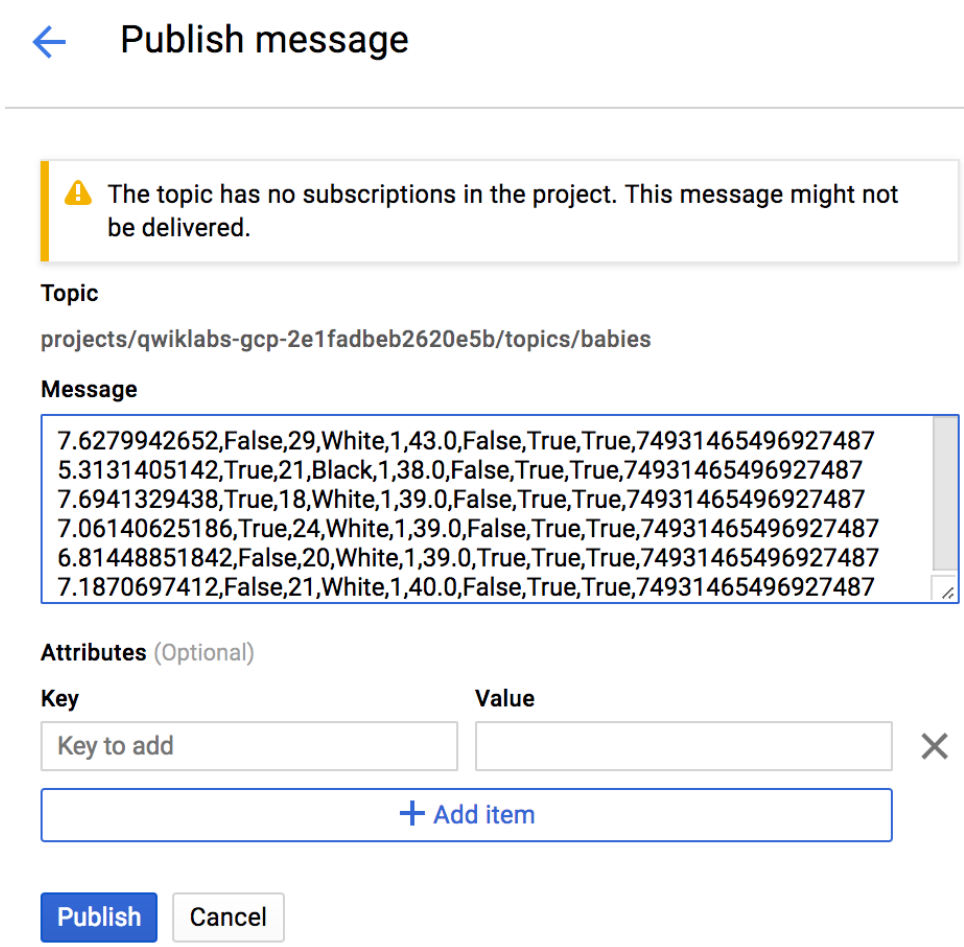
Step 5

Copy some lines from your example.csv.gz

```
cd ~/training-data-analyst/courses/machine_learning/deepdive/06_structured/serving
zcat exampledata.csv.gz
```

Step 6

On your GCP Console, go back into Pub/Sub, click on the **babies** topic, and then click on "Publish message" button on top. In the message box, paste the lines you just copied from exampledata.csv.gz and click on **Publish** button.



Step 7

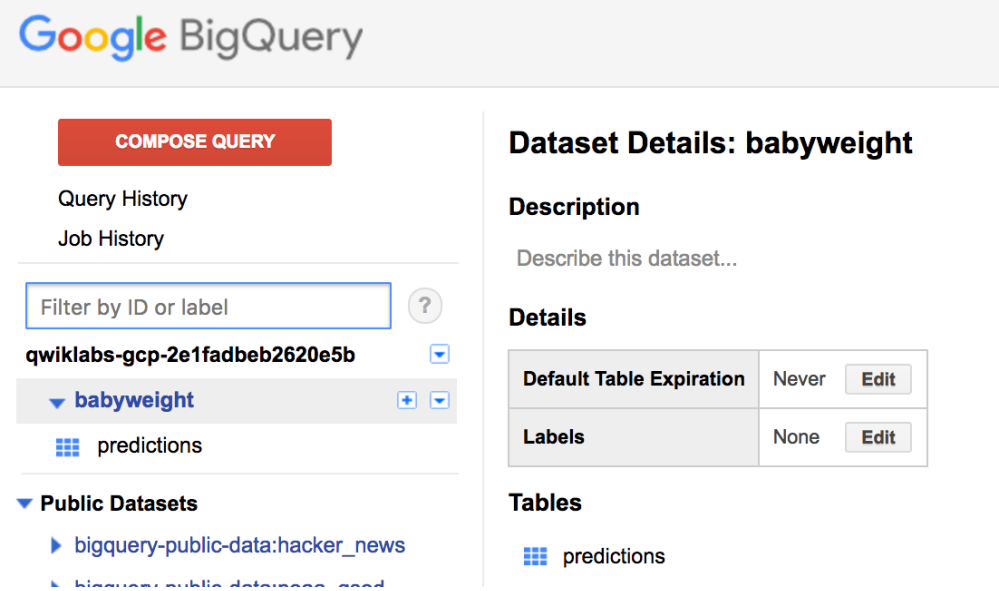
You may go back into Dataflow jobs on your GCP Console, click on your job and see how the run details have changed for the steps, for example click on write_toBQ and look at Elements added.

Step 8

Lets verify that the predicted weights have been recorded into the BigQuery table. On your GCP console, click on BigQuery. This typically opens a new tab and may ask for you qwiklabs account's

password. Once entered, you will be redirected to BigQuery console. Look at the left-side menu and you should see the **babyweight** dataset. Click on the blue down arrow to its left, and you should see your **prediction** table.

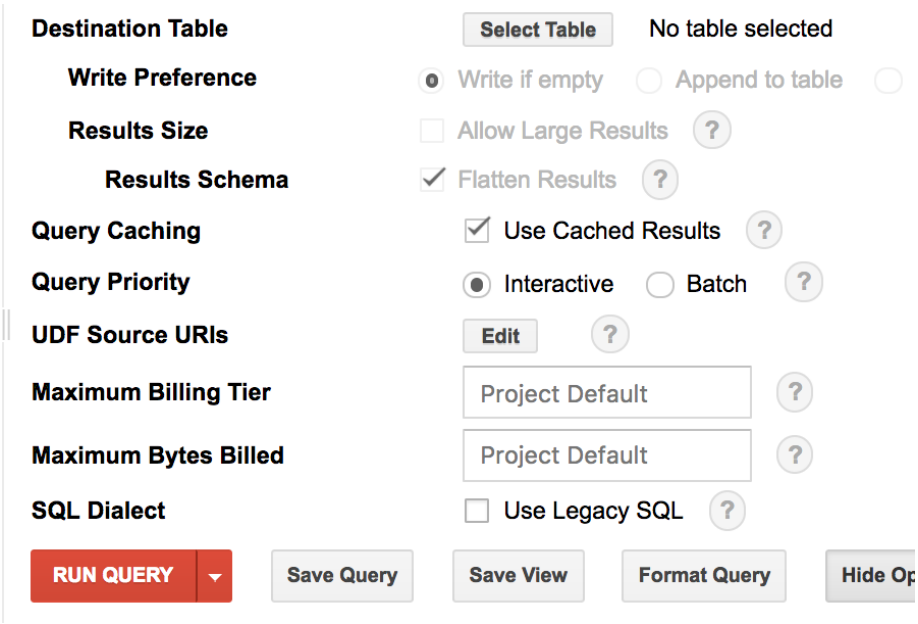
Note: If you do not see the prediction table, give it a few minutes as the pipeline has allowed-latency and that can add some delay.



Step 9

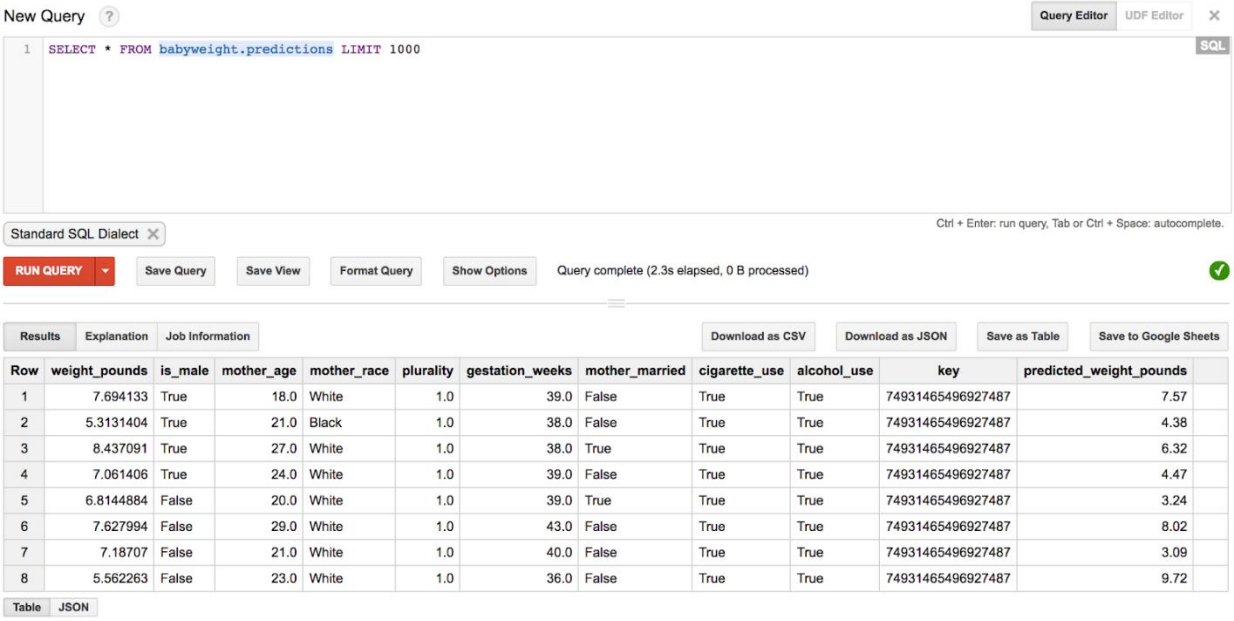
Click on Compose Query button on the top left. Type the query below in the query box to retrieve rows from your predictions table. Click on **Show Options** button under the query box and uncheck "**Use Legacy SQL**".

```
SELECT * FROM babyweight.predictions LIMIT 1000
```



Step 10

Click the Run Query button. Notice the **predicted_weights_pounds** column in the result.



Row	weight_pounds	is_male	mother_age	mother_race	plurality	gestation_weeks	mother_married	cigarette_use	alcohol_use	key	predicted_weight_pounds
1	7.694133	True	18.0	White	1.0	39.0	False	True	True	74931465496927487	7.57
2	5.3131404	True	21.0	Black	1.0	38.0	False	True	True	74931465496927487	4.38
3	8.437091	True	27.0	White	1.0	38.0	True	True	True	74931465496927487	6.32
4	7.061406	True	24.0	White	1.0	39.0	False	True	True	74931465496927487	4.47
5	6.8144884	False	20.0	White	1.0	39.0	True	True	True	74931465496927487	3.24
6	7.627994	False	29.0	White	1.0	43.0	False	True	True	74931465496927487	8.02
7	7.18707	False	21.0	White	1.0	40.0	False	True	True	74931465496927487	3.09
8	5.562263	False	23.0	White	1.0	36.0	False	True	True	74931465496927487	9.72

Step 11

Remember that your pipeline is still running. You can publish additional messages from your example.csv.gz and verify new rows added to your predictions table. Once you are satisfied, you may stop the Dataflow pipeline by going into your Dataflow Jobs page, and click the **Stop job** button on the right side Job summary window.

Job

Job summary

Job name	addprediction-gcpstaging28950student-0901173506-fd976478
Job ID	2017-09-01_10_35_26-15406689007149441589
Job status	Running Stop job
SDK version	Google Cloud Dataflow SDK for Java 2.0.0
Job type	Streaming
Start time	Sep 1, 2017, 10:35:27 AM
Elapsed time	1 hr 2 min

Autoscaling

Workers	1
Current state	Worker pool started.