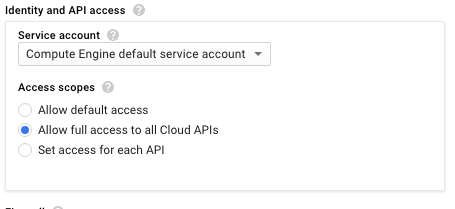
**Create Compute Engine**

## Create Compute Engine instance with API access

1. Go to **[Cloud Platform Console](https://console.cloud.google.com/" \t "_blank)**

2. Click on the Menu (three horizontal lines) and select **Compute Engine**

3. Click **Create Instance** and in the form, under Identify and API access 🡪 Access Scopes 🡪 **Allow full access to all Cloud APIs** 🡪 Click **Create**



## SSH into instance and Install Software

1. Once instance is created, click on **SSH**:

**Note**: SSH keys are automatically transferred and done directly from the browser, with no extra software needed.

2. To find some information about the Compute Engine instance, type the following into the command-line:

cat /proc/cpuinfo

3. Install Git on the machine with the below command:

sudo apt-get update sudo apt-get -y -qq install git

4. Verify that git is now installed

git --version

**Interact with Cloud Storage**

## Ingest Sample data

1. SSH into your Compute Engine instance (See previous section)

2. On the command-line, type the below to download the code from Github

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

3. Navigate to the folder corresponding to this lab:

cd training-data-analyst/CPB100/lab2b

4. Examine the ingest code using less:

less ingest.sh

The ingest.sh downloads a dataset of earthquakes in the past 7 days from the US Geological Survey to disk.

5. Run the ingest code and verify that some data has been downloaded using the head command.:

bash ingest.sh

head earthquakes.csv

## Transform the data

1. The transformation code is explained in detail in this notebook written in Datalab:

[https://github.com/GoogleCloudPlatform/datalab-samples/blob/master/basemap/earthquakes.ipynb](https://github.com/GoogleCloudPlatform/datalab-samples/blob/master/basemap/earthquakes.ipynb" \t "_blank)

2. First, install the necessary Python packages on the Compute Engine instance:

bash install\_missing.sh

3. Then, run the transformation code and check the directory for a new image file

python transform.py

ls -l

## Create bucket

1. Go to **[Cloud Platform Console](https://console.cloud.google.com/" \t "_blank)** 🡪 From top left menu select **Storage** 🡪 **Create Bucket** 🡪 Choose a globally unique bucket name (your project name is unique, so you could use that) 🡪 Click **Create**.

3. Note down the name of your bucket: <YOUR-BUCKET>.

## Store data

1. From the SSH window of the Compute Engine instance, type the below to copy the files to Cloud Storage

gsutil cp earthquakes.\* gs://<YOUR-BUCKET>/

2. From GCP console, click on your bucket name, and notice there are three new files present

## Publish Cloud Storage files to web

1. Select all three earthquakes files that you uploaded to the bucket and click on **Share publicly**

2. Click on the **Public link** checkbox corresponding to **earthquakes.htm, note the URL of the Bucket**

## Clean up

1. Go to **[Cloud Platform Console](https://console.cloud.google.com/" \t "_blank)** 🡪 From top left menu select **Compute Engine** 🡪 Click on the checkbox corresponding to the instance that you created 🡪Click the **Delete** button

**Create & Populate Cloud SQL**

Explore Data in Cloud Shell

1. From GCP console, start a local **Cloud Shell**

2. On the command-line, type below to download code from github

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

3. Navigate to the folder and examine SQL for table creation

cd training-data-analyst/CPB100/lab3a

less cloudsql/table\_creation.sql

|  |  |
| --- | --- |
| Table Name | Columns |
| Accommodation | Id, title, location, price, rooms, rating, type |
| Rating | userId, accoId, rating |
| Recommendation | userId, accoId, prediction |

4. Examine the data files using head:

head cloudsql/\*.csv

## Stage .sql and .csv files into Cloud Storage

1. From Cloud Shell within the lab3a directory, type the below to copy to Cloud Storage

gsutil cp cloudsql/\* gs://<BUCKET-NAME>/sql/

2. From the GCP console 🡪 Storage, navigate to your bucket and verify that the .sql and .csv files now exist

Create Cloud SQL instance

1. From the GCP console 🡪 select **SQL** (in the Storage section) 🡪 **Create Instance** 🡪 Choose **MySQL**, click Next. 🡪 Choose Cloud SQL **Second Generation** 🡪 Name the instance ID **rentals**

2. Scroll down and specify a root password. Before you forget, note down the root pw in a table like this.

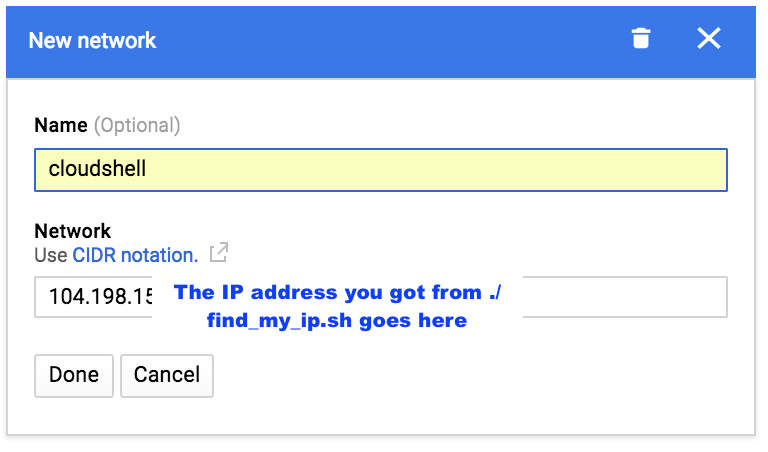
|  |  |
| --- | --- |
| Machine | Password/IP Address |
| Root password | \_\_\_\_\_\_? |
| Cloud Shell (client) | (Step 4) |
| Cloud SQL (MySQL server) | (Step 7) |

3. Scroll down and under **Authorize networks** click on the **Add network** button

4. From Cloud Shell within the lab3a directory, find your IP address by typing:

bash ./find\_my\_ip.sh

5, In the Add Network box, plug your IP address (Name can be anything) and click **Done**



**Note**: If you lose your Cloud Shell VM due to inactivity, you will have to reauthorize your new Cloud Shell VM with Cloud SQL. For your convenience, lab3a includes a script called **authorize\_cloudshell.sh** that you can run.

6.Click **Create** 🡪 It will take a minute or so for your Cloud SQL instance to be provisioned.

**Step 10**

Note down the IP address of your Cloud SQL instance (from the browser window) in the *third* row of the table you started:

Note this down in the second row of a table like this:

|  |  |
| --- | --- |
| Machine | Password/IP Address |
| Root password | (Step 5) |
| Cloud Shell (client) | (Step 7) |
| Cloud SQL (MySQL server) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ? |

## Create tables

To import table definitions from Cloud Storage:

## ****Step 1****

Click on the hyperlink named **rentals**i.e. your Cloud SQL instance name

## ****Step 2****

Click on **Import**(on the top menu bar)

## ****Step 3****

Click on the **Browse** button and browse to table\_creation.sql and **Select** it

## ****Step 4****

Click **Import**.

## Populate tables

To import CSV files from Cloud Storage:

## ****Step 1****

From the GCP console page with the Cloud SQL instance details, click on **Import** (top menu)

## ****Step 2****

Click on the **Browse** button, browse to accommodation.csv and **Select** it. Fill out the rest of the dialog as follows:

The Database is recommendation\_spark

The Table is **Accommodation**

## 

## ****Step 3****

Repeat the **Import** for rating.csv, changing the table name to **Rating**.

## Explore Cloud SQL

To explore Cloud SQL, you can use the tool mysql from Cloud Shell:

## ****Step 1****

In the Cloud Shell window, type:

mysql --host=<MySQLIP> --user=root --password

The IP address is the one for the database server (i.e. the third row in the notes). You can also find it from the instance details on the cloud console.

MySQL will prompt you for the root password. Type that into the prompt when prompted.

## ****Step 2****

In the Cloud Shell, at the mysql prompt, type:

use recommendation\_spark;

This sets the database in the mysql session.

**Step 3**

View the list of tables you created. This will be helpful to prevent any typos in your query in step 4.

show tables;

## Step 4

Let's see if there is a great deal out there somewhere:

select \* from Accommodation where type = 'castle' and price < 1500;

**Create Dataproc Clusters**

Dataproc Intro

1. Dataproc is a managed service for creating clusters of computers that can be used to run Hadoop and Spark applications. Dataproc clusters are pre-configured with software commonly used in Hadoop ecosystems like Python, Java, PySpark, Pig and Hive. Dataproc clusters are also pre-configured with HDFS.
2. Dataproc clusters can easily be created in just a couple minutes and clusters can be easily configured to run jobs both big and small. Because clusters can be created so quickly, they can also be deleted as soon and jobs are complete. With Google's per-minute billing, this allows jobs to be run at a minimal cost.
3. Dataproc requires no upfront payment. You only pay for the resources used for the time the clusters are running.

Create a Dataproc Cluster

How to create a cluster from GCP Web Console

1. Go to **[Cloud Platform Console](https://console.cloud.google.com/" \t "_blank)**

2. Click left hand menu 🡪 Click **Compute Engine**.

This ensures that necessary fraud checks are carried out and APIs are enabled. It will preload and reduce the wait times associated with later steps too.

3. Click left hand menu 🡪 Click **Dataproc 🡪** Click **Create cluster**

Note: If you get an "Enable API" popup, go ahead and click Enable.

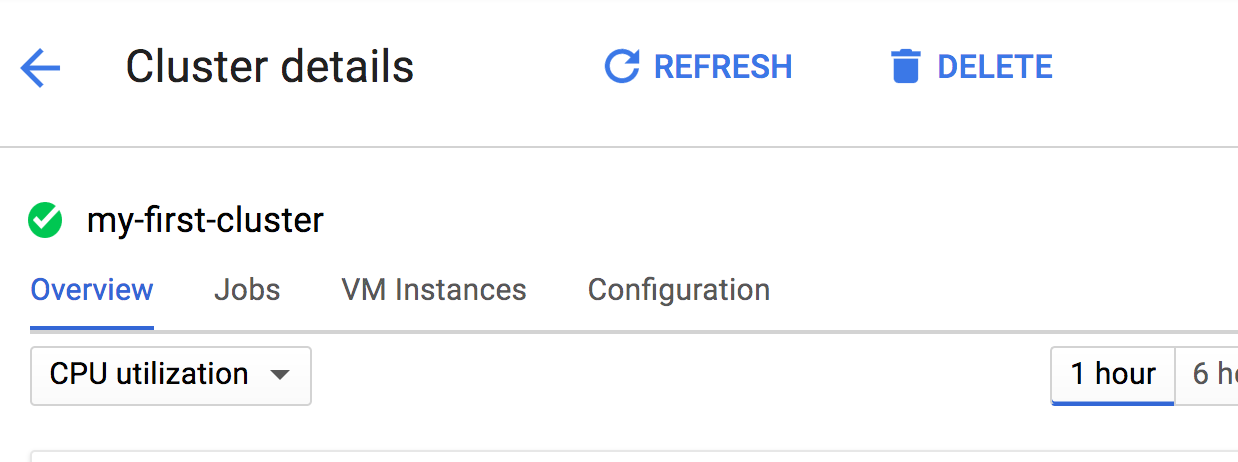
4. Create the smallest possible cluster with below configuration 🡪 Click **Create**🡪 May take a couple of minutes

* Name: **my-first-cluster**
* Zone: **us-central1-a**
* Master node Machine type: **n1-standard1 (1vCPU, 3.75GB memory)**
* Master node Primary disk size: **10 GB**
* Worker nodes Machine type: **n1-standard1 (1vCPU, 3.75GB memory)**
* Worker nodes: **2**
* Work nodes primary disk size: **10 GB**

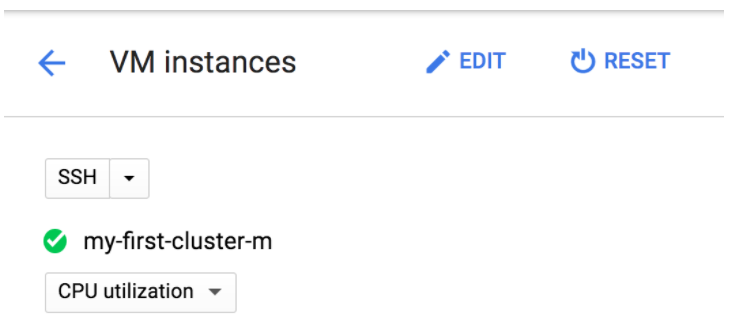
Access the Cluster Master with SSH

How to SSH into master node and discover what programs are installed

1. When you see green check next to created cluster 🡪 Click Cluster name to open **Cluster details 🡪** Click **VM Instances** tab to see a list of machines in your cluster 🡪 Click on the master node (**my-first-cluster-m**)



2. Click **SSH** button to connect to master node machine in a new terminal



3. Type below commands to confirm program versions already pre-installed on the machine.

python --version

java -version

scala -version

pyspark --version

pig --version

hive --version

## Accessing the Cluster Master with the Browser

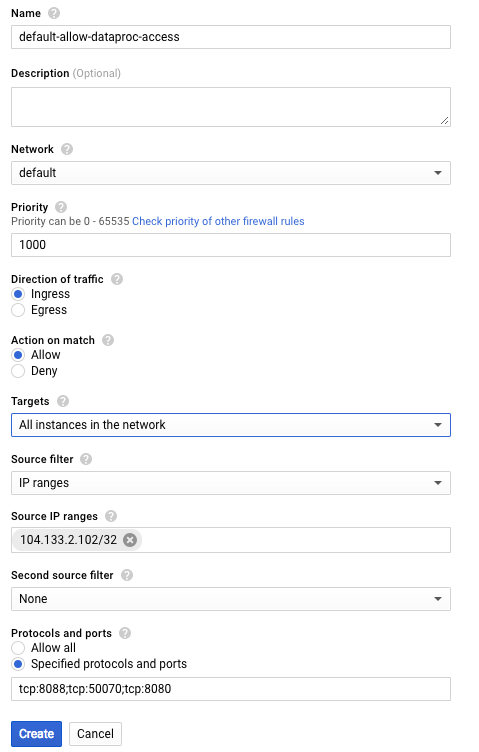
1. Go to **[Cloud Platform Console](https://console.cloud.google.com/" \t "_blank)**

2. Click left hand menu 🡪 Click **Networking**

3. You are going to allow access to your Dataproc cluster, but only to your machine. To do this, you will need to know your IP Address. Go to this URL to find your local IP address: [http://ip4.me/](http://ip4.me/" \t "_blank)

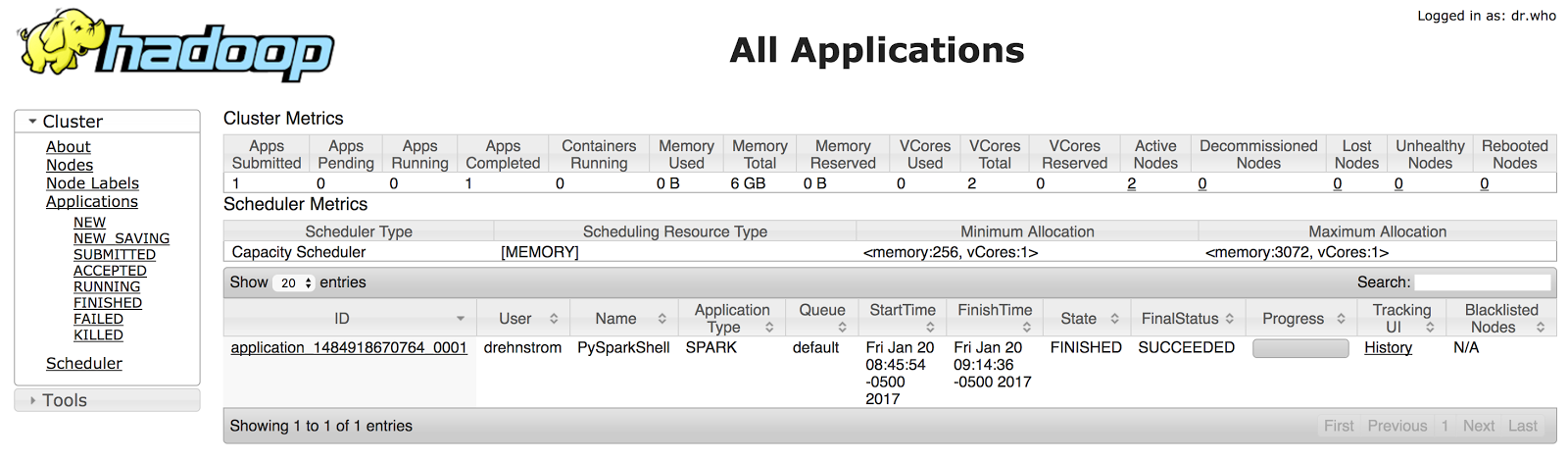
**4. On left hand menu 🡪 Firewall rules** 🡪 Click **Create Firewall Rule** 🡪 Enter the below 🡪 Click **Create**

* Name: **default-allow-dataproc-access**
* Targets: **All instances in the network**
* **Source filter:** **IP ranges**
* Source IP ranges: **Enter your ip address followed by /32. E.g. 1.2.3.4/32**
* P**rotocols and ports**: **Specified Protocols and ports** 🡪 **tcp:8088;tcp:50070;tcp:8080**

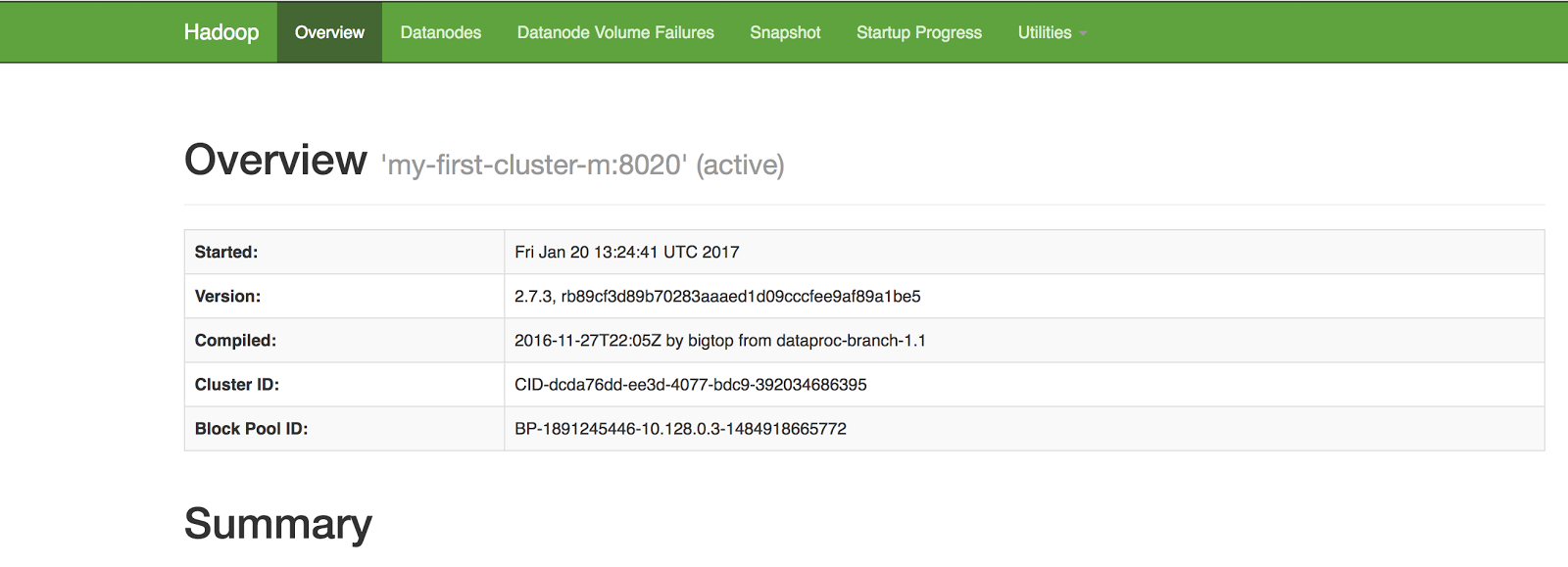
**Note**: This firewall rule allows access to tcp port 8088 (Hadoop) 50070 (HDFS) and 8080 (Datalab). We will install Datalab later

5. Go back to the **Dataproc** 🡪 Click Cluster name to open **Cluster details 🡪** Click **VM Instances** tab 🡪 Click **master node 🡪** Scroll down to see master node's **External IP** address, and copy it to your clipboard.

**Note**: All the nodes in the Dataproc cluster are really Compute Engine virtual machines. You could also find the master node's IP address from **Products and Services** 🡪 Compute Engine 🡪 Find your master node named my-first-cluster-m 🡪 Copy the external IP address.

6. Open new browser tab and paste in the master node's IP address, followed by the port **8088** to access your Hadoop. E.g. 1.2.3.4:8088 🡪 Should open page like below 🡪 Click various links on left to explore

7. Open another new browser tab and paste in the master node's IP address, followed by the port **50070** to access your HDFS cluster similar to below. Explore this as well.



8. Close the Hadoop and HDFS browser tabs. Go back to the window with the console and close it as well.

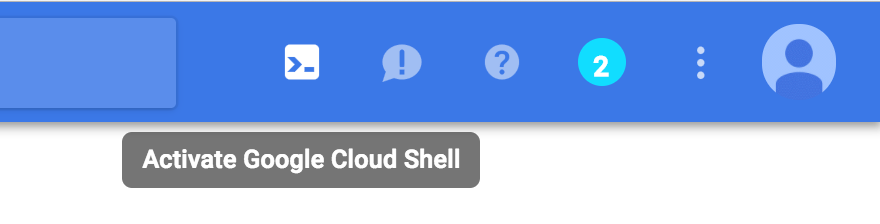
9. In Cloud Platform Console, return to the **Dataproc** 🡪 Select the checkbox next to your cluster 🡪 **Delete**

Managing Clusters with the CLI

You will now create a cluster using the command line interface

1. Go to **[Cloud Platform Console](https://console.cloud.google.com/" \t "_blank)**

2. Click left hand menu 🡪 Click **Dataproc**.

3. Click **Activate Google Cloud Shell**icon on upper right side of toolbar 🡪 Cloud Shell terminal window

4. Paste the following command into cloud shell and hit Enter 🡪 This creates a Dataproc cluster named **my-second-cluster** in the **us-central1-a** zone. It creates a master node with 1 CPU and a 50 GB disk and 2 worker nodes with the same resources. 🡪 Notice Dataproc home screen it is being created

gcloud dataproc clusters create my-second-cluster --zone us-central1-a \

--master-machine-type n1-standard-1 --master-boot-disk-size 50 \

--num-workers 2 --worker-machine-type n1-standard-1 \

--worker-boot-disk-size 50

5. When the green check appears next to **my-second-cluster**, click on the cluster and explore its details.

* What machine type is used for the master and worker nodes? **n1-standard-1**
* How many workers nodes were created? **2**

6. Paste below command into cloud shell and execute. This command deletes the cluster you just created. When prompted, confirm that you want to delete your cluster.

gcloud dataproc clusters delete my-second-cluster

7. After your cluster is deleted and not visible in Web console (you may need to click *Refresh*) 🡪 Click **Create cluster** 🡪 Enter the below settings 🡪 **Do not** click the Create button.

* Name: **my-third-cluster**.
* Zone: **us-central1-b**
* Master node machine type: **n1-standard1 (1vCPU, 3.75GB memory)**
* Worker nodes machine type: **n1-standard1 (1vCPU, 3.75GB memory)**
* Worker nodes: **2**

Below the Create and Cancel buttons, click the link which reads **command line** 🡪 Window pop-up appears with the command using the settings you've specified 🡪 Copy command to the clipboard 🡪 Paste command into Cloud Shell and execute 🡪 Click **Cancel** button on the ***Create a cluster***page 🡪 Notice another cluster is being created.

8. When the cluster is done spinning up, explore its details to confirm it’s as expected.

9. From Console 🡪 Click **Compute Engine** 🡪 Notice the master and worker nodes are really Compute Engine VMs

10. From Console 🡪 Click **Dataproc** 🡪 Delete the Cluster

**Run Spark and Pig Programs**

## Setup Firewall Rule (See above)

## Creating a Dataproc Cluster with Cloud Storage Bucket

1. From [Cloud Console](https://console.cloud.google.com/" \t "_blank) c lick on Dataproc, then click on the Enable API button in the window that pops up.

2. Open a **Google Cloud Shell,**enter the following command to create a cluster:

gcloud dataproc clusters create my-cluster --zone us-central1-a --master-machine-type n1-standard-1 --master-boot-disk-size 50 --num-workers 2 --worker-machine-type n1-standard-1 --worker-boot-disk-size 50 --network=default

**3. In Google Cloud Shell,**enter the following command to create a Cloud Storage bucket with the same name as your project ID in the same region as your cluster. Both Cloud Storage buckets and Project ID's have to be unique, so unless you are very unlucky your project ID would not have been previously used for a bucket name.

gsutil mb -c regional -l us-central1 gs://$DEVSHELL\_PROJECT\_ID

4. Use the menu in the Web Console to navigate to the **Storage** service. Confirm that your bucket was created.

## Copy Code and Files to Cloud Storage Bucket

**1. Open Google Cloud Shell**and enter the commands below to copy some pre-created files into your bucket (make sure to plug in your bucket name).

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

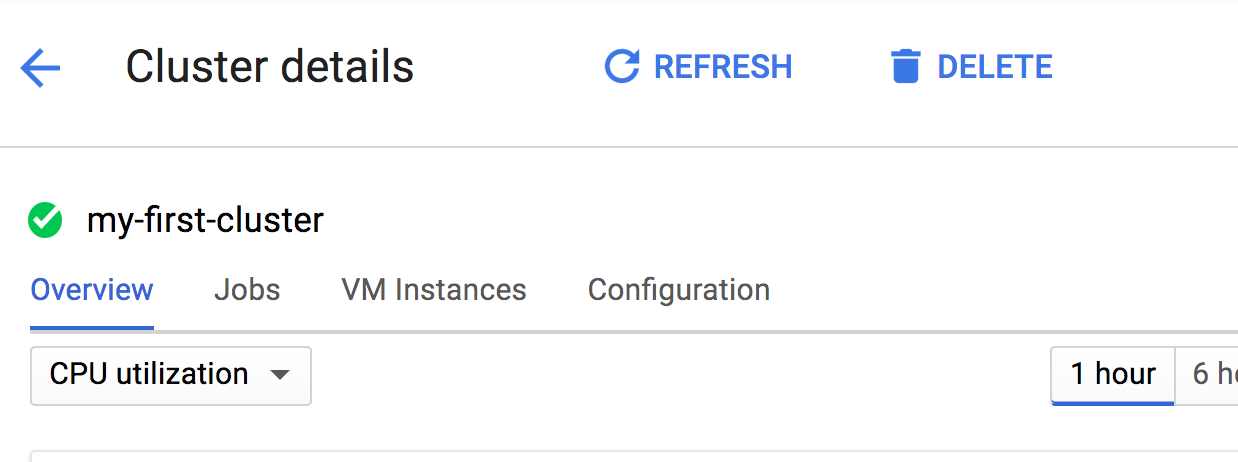
cd training-data-analyst/courses/unstructured

./replace\_and\_upload.sh <YOUR-BUCKET-NAME>

## Developing using PySpark REPL

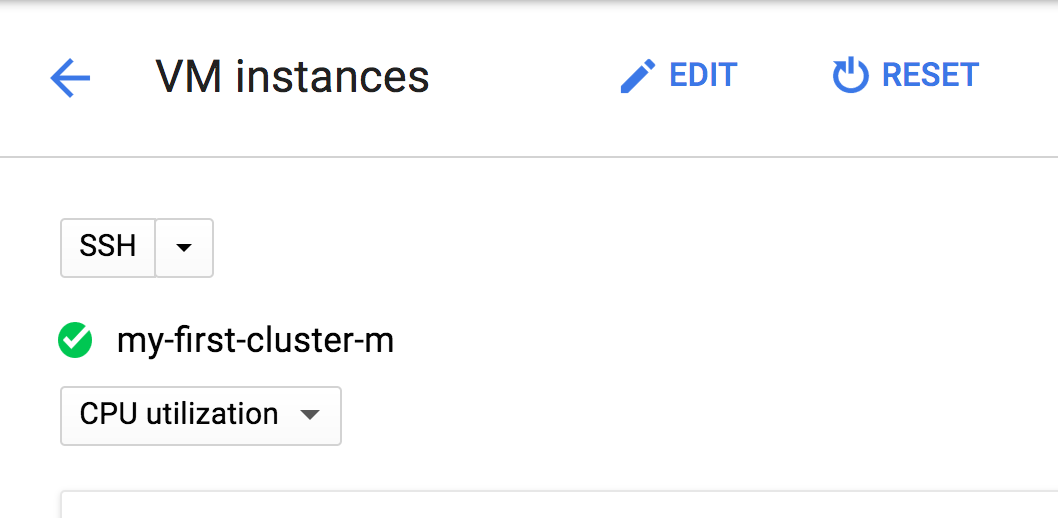
You will SSH into the master node and run the Python Spark Read-Evaluate-Process-Loop (REPL) interpreter.

1. Navigate to your Dataproc cluster and click on the cluster name. This opens the **Cluster details** page.



2. Click the **VM Instances** tab to see a list of machines in your cluster. Click on the master node (**my-cluster-m**), to see that machine's details.

3. Click the **SSH** button to connect to that machine. This will open a new window or tab in your browser with a terminal window that is connected to your master node machine.



**4. Type pyspark** at the command prompt to open the PySpark shell.

5. Type the following code to run a simple PySpark job.

data = [0, 1, 2, 3, 4, 5] # range(6)

distData = sc.parallelize(data)

squares = distData.map(lambda x : x\*x)

res = squares.reduce(lambda a, b : a + b)

print res

6. Exit Spark by typing:

quit()

7. While you could develop and run PySpark programs using the REPL, a more common way to develop PySpark programs is to use a Python notebook, and a more common way to execute PySpark programs to submit a Python file. You will do both of these in subsequent sections and labs.

## Pig Job that reads from HDFS

You will now execute a Pig job and view its results. You will also use the HDFS cluster provided by Google Cloud Dataproc

1. If you don't have the SSH terminal to the cluster master still available, navigate to the **Dataproc** service in the Web console and click on the **Clusters** link. Click on your cluster (it should be named**my-cluster**) to see its details, then click the **VM Instances** tab, and then click on the master node to view its details. Finally, click the **SSH** button to connect to the master.

2. Enter following command to create a directory for this exercise and move into it:

mkdir lab2

cd lab2

3. Enter following command to copy a data file and a pig script into the folder you just created. Make sure to plug in your actual bucket name.

gsutil -m cp gs://<YOUR-BUCKET-NAME>/unstructured/pet-details.\* .

Two files were copied from Cloud Storage to the cluster. You can view them by entering the following commands.

cat pet-details.txt

This just shows a simple data file we will copy into HDFS and then transform using Pig. Enter the following command to see the Pig script you will run, and take a minute to study it.

cat pet-details.pig

4. Copy the text file into HDFS. Use the following code.

hadoop fs -mkdir /pet-details

hadoop fs -put pet-details.txt /pet-details

5. From Web console check details of your master node. Find the master node's external IP address and copy it to the clipboard. Then, open a new tab in your browser, paste in the IP address with suffix **:50070**. This will open the Hadoop management site. From the **Utilities** menu on the right select **Browse the file system**.Verify that you have a folder called **pet-details** and inside it you should have a file called **pet-details.txt**.

6. In SSH window, run the following command to run Pig:

pig < pet-details.pig

Click **Submit** to start the job. It will take about a minute to run. Wait until it completes.

7. Go back to the tab with the Hadoop management site and again browse the file system. The output from this Pig job should be in a folder called **GroupedByType**. If you look in that folder you should see a file named **part-r-00000**.

8. Let’s look at the output file.

First you have to get the file off the HDFS file system. Go back to your SSH session where you are connected to the master node. You should currently be in the folder **lab2**. Make a directory below it and move into by entering the following commands.

mkdir ~/lab2/output

cd ~/lab2/output

9. Enter the following command to get the output file from HDFS and copy it into this folder, then view results

hadoop fs -get /GroupedByType/part\* .

cat \*

Compare the original data file, the Pig script and the final output. Try to figure out why the output is the way it is.

**Run Dataproc Jobs**

## Create a Dataproc Cluster (See above)

## Submitting PySpark Jobs

In previous lab, you ran code after logging into the cluster. In the case of Pig, you copied data over to the cluster's HDFS before you ran it. In this section, you will submit a Spark job and view its results without copying anything (code or data) to the cluster.

1. From [Cloud Console](https://console.cloud.google.com/" \t "_blank), navigate to Storage and click on your bucket. It should have some files in the unstructured folder. Click on the file, lab2-input.txt and view its contents. This file contains a comma separated list of keys and values.

Also view the contents of the file, lab2.py. This is a PySpark job that organizes the input file by key and the total number for each type of pet. Notice that both the code and data are on Cloud Storage. We have not copied either of these to the cluster.

2. Navigate to the **Dataproc** service in the Web Console.

3. On left-hand navigation pane select **Jobs**. Then click the **Submit job** button.

4. Fill out as:

Cluster: **my-cluster**

**Job type: PySpark**.

**Main python file:**Enter the path to the PySpark file lab2.py that is in your bucket. It should be in the form shown below, but replace <YOUR-BUCKET-NAME> with the name of your bucket: gs://<YOUR-BUCKET-NAME>/unstructured/lab2.py

5.Click **Submit** button at the bottom of the form.

6. Wait for job to succeed and then click on Job ID to see details. Take a look at job output to see the results.

7. To run the job again click the **Clone** button and the top, then **Submit** the job a second time.

8. To run the job using the CLI, go back to the Google Cloud Shell and paste in the following command replacing <YOUR-BUCKET-NAME> with the name of your bucket.

gcloud dataproc jobs submit pyspark --cluster my-cluster gs://<YOUR-BUCKET-NAME>/unstructured/lab2.py

**Leverage GCP Services**

## Customizing and Installing Software on Dataproc Clusters

Additional software can be added to Dataproc clusters, and clusters can be customized using initialization actions. Initialization actions are simply executables that are run when the cluster is being created. You will use a pre-built initialization action to install Datalab and a custom one to install the Google Client Python API. Datalab allows you to write interactive Python and PySpark notebooks that are useful in data analysis. You will create a couple of notebooks in this exercise that make use of our Dataproc cluster and also integrate with Google BigQuery and Google Cloud Storage.

## 1a. Write executable program that runs as root, OR

#!/bin/bash

apt-get update || true

apt-get install –y python-numpy python-scipy python-matplotlib python-pandas

1b. Run code only on Master Node, or only on Worker Nodes

#!/bin/bash

apt-get update || true

ROLE=$(/usr/share/google/get\_metadata\_value attributes/dataproc-role)

If [[ “${ROLE}” == ‘Master’ ]]; then

apt-get install –y vim

else

#something that goes only on worker

Fi

#things that go on both master and worker

apt-get install –y python-numpy python-scipy python-matplotlib python-pandas

2. Upload it to Google Cloud Storage

gsutil cp my\_init.sh gs://<mybucket>/init-actions/my\_init.sh

A library of pre-built initialization actions are hosted in this publicly-accessible Google Cloud Storage Bucket:

gsutil ls gs://dataproc-initialization-actions

<https://github.com/GoogleCloudPlatform/dataproc-initialization-actions>

3. Specify GCS location in Dataproc creation command

gcloud dataproc clusters create mycluster \

--initialization-actions gs://<mybucket>/init-actions/my\_init.sh \

--initialization-action-timeout 3m

## Creating an Initialization Action

You will create a custom initialization action to install a Python package.

1. From [Cloud Console](https://console.cloud.google.com/" \t "_blank), open a Google Cloud Shell and enter the following command to create a Cloud Storage bucket with the same name as your project ID

gsutil mb -c regional -l us-central1 gs://$DEVSHELL\_PROJECT\_ID

2. In **Cloud Shell,** git clone the course repository, and upload the custom initialization script to GCS. Change the bucket name as necessary.

git clone https://github.com/GoogleCloudPlatform/training-data-analyst cd training-data-analyst/courses/unstructured/ bash replace\_and\_upload.sh <YOUR-BUCKET-NAME>

3. View the custom initialization script. Change the bucket name as necessary.

gsutil cat gs://<YOUR-BUCKET-NAME>/unstructured/init-script.sh

#!/bin/bash

# install Google Python client on all nodes

apt-get update

apt-get install -y python-pip

pip install --upgrade google-api-python-client

ROLE=$(/usr/share/google/get\_metadata\_value attributes/dataproc-role)

if [[ "${ROLE}" == 'Master' ]]; then

echo "Only on master node ..."

fi

Creating a Dataproc Cluster with an Initialization Action

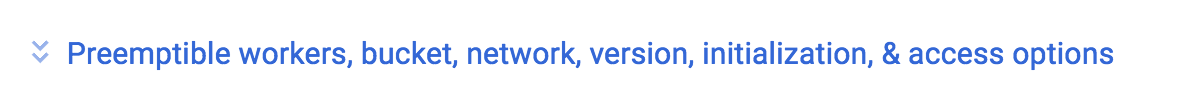
You will create a cluster that will include two initialization actions: (1) a pre-built action from Google to install Datalab, and (2) a custom initialization action to install a Python package.

1. Use the **Products and Services** menu to navigate to the **Dataproc** service. Click to *Enable API*. If you have any clusters currently running, you can delete them.

2. Click the **Create cluster** button and set the following parameters.

* Name your cluster **my-cluster**.
* Select the zone **us-central1-a**.
* In the **Master node | Machine type** drop-down select the first machine **n1-standard1(1vCPU, 3.75GB memory)**.
* In the **Worker nodes | Machine type** drop-down also select the first machine **n1-standard1(1vCPU, 3.75GB memory)**.
* Leave the number of worker nodes at the default of 2.

Click on the link shown below to expand more options.



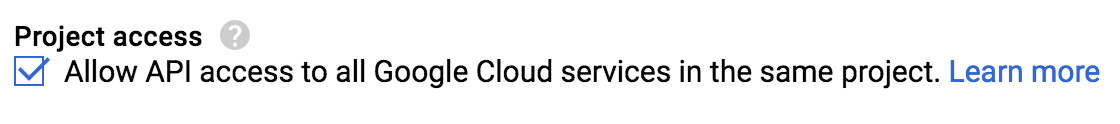
Copy and paste the following script URL into the **Initialization actions** text box and press **Enter**. (*This script installs Google Cloud Datalab on your cluster's master node*.)

gs://dataproc-initialization-actions/datalab/datalab.sh

Copy and paste this second initialization action into the **Initialization actions** text box and press **Enter**. Change the bucket name appropriately. (*This script installs the Google Python Client API on all the machines in the cluster and clones the course repository to the Master node, so that Datalab cab access notebooks that are in the repository.*)

gs://<YOUR-BUCKET-NAME>/unstructured/init-script.sh

Check the **Project access** box as shown below to allow your cluster to access other Google Cloud Platform services.



3. To create the cluster, either click the **Create** button or click on the **Command line** link and copy the command onto your clipboard and then run it from Google Cloud Shell.

gcloud dataproc clusters create my-cluster --zone us-central1-a \

--master-machine-type n1-standard-1 --master-boot-disk-size 50 \

--num-workers 2 --worker-machine-type n1-standard-1 \

--worker-boot-disk-size 50 --network=default \

--scopes 'https://www.googleapis.com/auth/cloud-platform' \

--project $DEVSHELL\_PROJECT\_ID \

--initialization-actions 'gs://dataproc-initialization-actions/datalab/datalab.sh','gs://<YOUR-BUCKET-NAME>/unstructured/init-script.sh'

4. It will take a little longer for your cluster to be created this time, because the scripts have to run. While you are waiting, browse to the following github site where you will find many other pre-written initialization actions.

[https://github.com/GoogleCloudPlatform/dataproc-initialization-actions](https://github.com/GoogleCloudPlatform/dataproc-initialization-actions" \t "_blank)

**Note:** Initialization actions are really just executables that run when a cluster is being creating. They are used to install additional software or customize your cluster as required by your programs. You can include one or more initialization actions when creating Dataproc clusters.

5. You are going to allow access to your Dataproc cluster, but only to your machine. To do this, you will need to know your IP Address. Go to the following URL to find out what it is: [http://ip4.me/](http://ip4.me/" \t "_blank)

6. From Cloud Console, click the menu on the left and select **Networking** from the Compute section. Click **Firewall rules** in the left-hand navigation pane. Click on the **Create Firewall Rule** button. Then, Enter the following:

* Name the rule **default-allow-dataproc-access**.
* Select **IP ranges** from the **Source filter** dropdown.
* In the source IP ranges text box enter your ip address followed by /32. So if your IP address is 1.2.3.4 then the text box would read 1.2.3.4/32.
* For **Targets**, select "All instances in the network"
* For **Protocols and ports**, select "Specified Protocols and ports", and enter the following in the text box:

tcp:8088;tcp:50070;tcp:8080

7. Once your cluster is finished initializing, click on its name to go to its details page, then click on the VM Instances tab, and finally click on the master node to view its details. Scroll down and find the master node's external IP address and copy it to your clipboard.

Open a new browser tab, paste in this IP address and then add **:8080** after the address. This opens Datalab. You will be redirected to the Datalab main screen as shown below:



**Important:**The reason you can browse to this port is because earlier you created a firewall rule that opened port 8080 to your machine. Be careful when opening this port. You would not want to create a rule that allowed everyone access to Datalab. If you did, then your cluster could be easily hacked.

## Creating a Simple Datalab Notebook

Let's just create a simple Python Notebook and make sure everything is working.

1. On left side of the Datalab home page click the **+ Notebook** button.

2. In the first cell, just enter the following Python code.

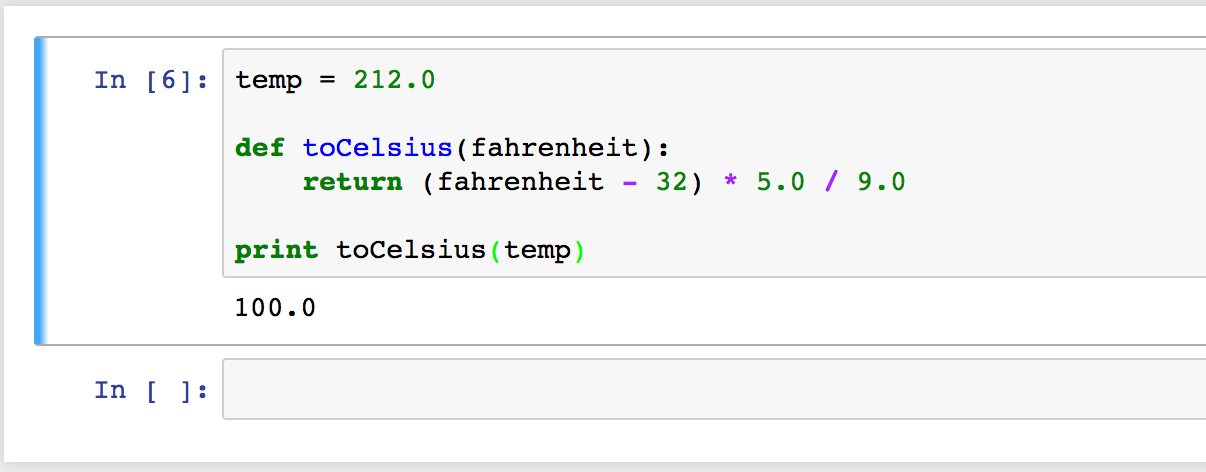
temp = 212.0

def toCelsius(fahrenheit):

return (fahrenheit - 32) \* 5.0 / 9.0

print toCelsius(temp)

3. Click the **Run** button in the toolbar and examine the results. It should look as shown below. (It might take a little while for the notebook to start.)



## Running a BigQuery SQL

The Python package Pandas comes with support to run BigQuery queries.

1. In the second code block add the following code and click **Run**. These import statements will allow you to run a BigQuery query.

import pandas as pd

from pandas.io import gbq

print "Imports run."

2. In the next code block, add the following code changing the **projectId** variable to your project id.

(You can find your project id in the Google Cloud Platform Web Console. Select Home from the Cloud Console menu.)

projectId = "YOUR-PROJECT-ID-HERE" # CHANGE

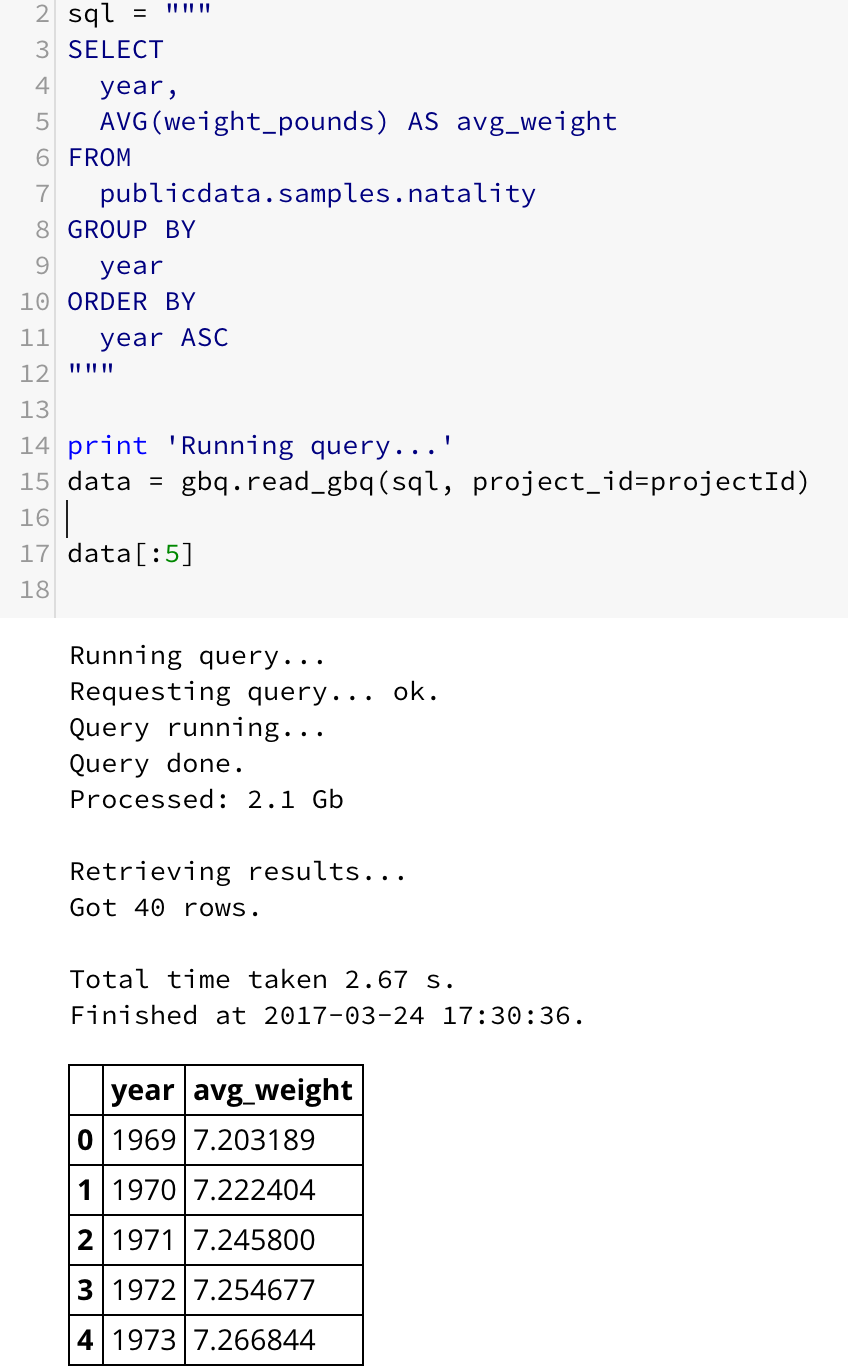
sql = """ SELECT year, AVG(weight\_pounds) AS avg\_weight FROM publicdata.samples.natality GROUP BY year ORDER BY year ASC """

print 'Running query...'

data = gbq.read\_gbq(sql, project\_id=projectId)

data[:5]

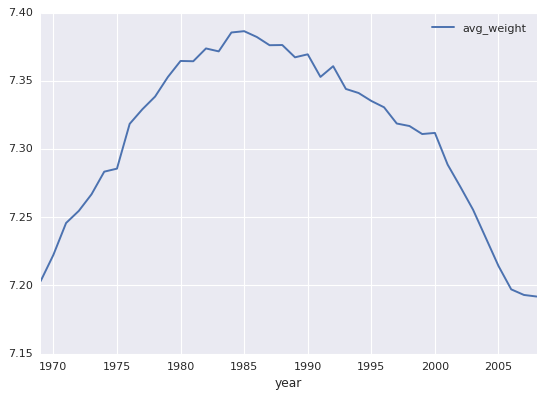
Click the **Run** button. The BigQuery query is run and the results put into a Pandas DataFrame. The last line just outputs the first 5 records. The results are shown below



3. In the next code block, add the following code to plot a graph using Pandas

data.plot(x='year', y='avg\_weight');

You should get a graph that looks like this:



4. In the Datalab menu bar, select **Notebook | Rename**. Name the notebook **BigQuery-Test** and then click OK. You can then close that tab and return to the Datalab Home page.

5. Back at the Datalab home page in the upper right corner of the toolbar are 4 icons. Hover over the second one (the one that looks like a stack of progress bars) and the resulting tooltip should read **Running Sessions**. Click on that icon.



On the resulting page you should see one active notebook, the BigQuery-Test notebook you just created.

Click the **Shutdown** button on the right side and then close this tab.

## Using PySpark in a Datalab Notebook

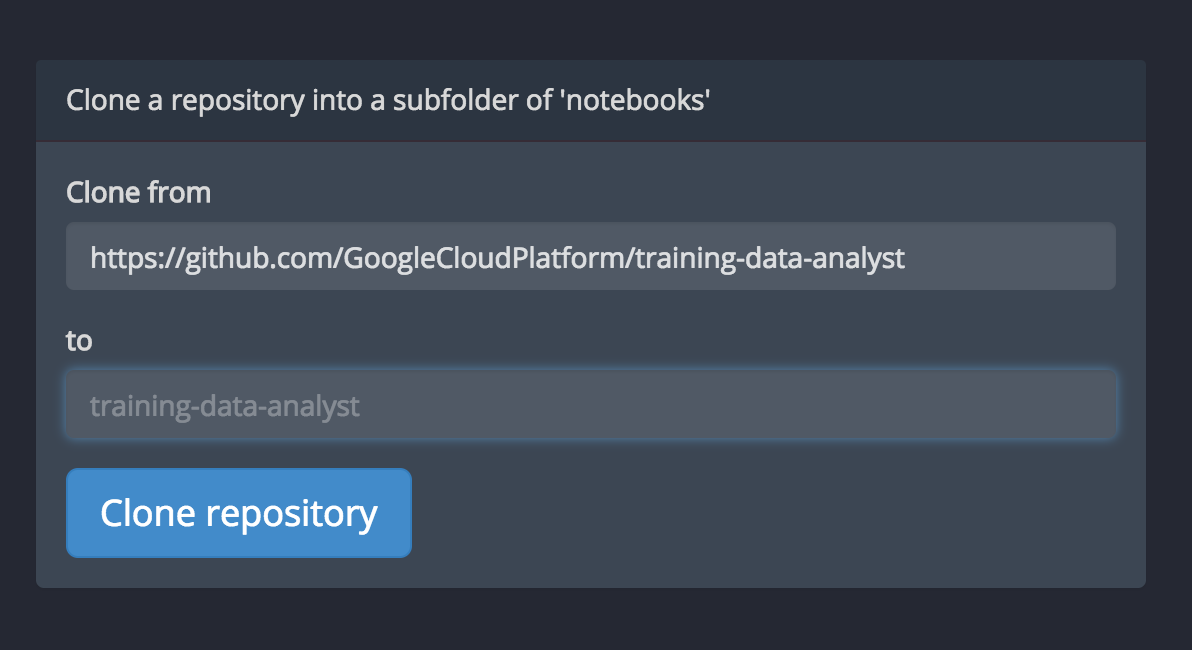
The last notebook didn't run anything in parallel on your Dataproc cluster. This time, let's get a notebook from the GitHub repository and execute it. This notebook uses PySpark and makes use of your Spark cluster.

1. Back at the Datalab home page in the upper right corner of the toolbar are 4 icons. Hover over the first one (the one that looks like a fork in the road) and the resulting tooltip should read **Open ungit**. Click on that icon.



2. Fill out the form to clone the github repository corresponding to the course:

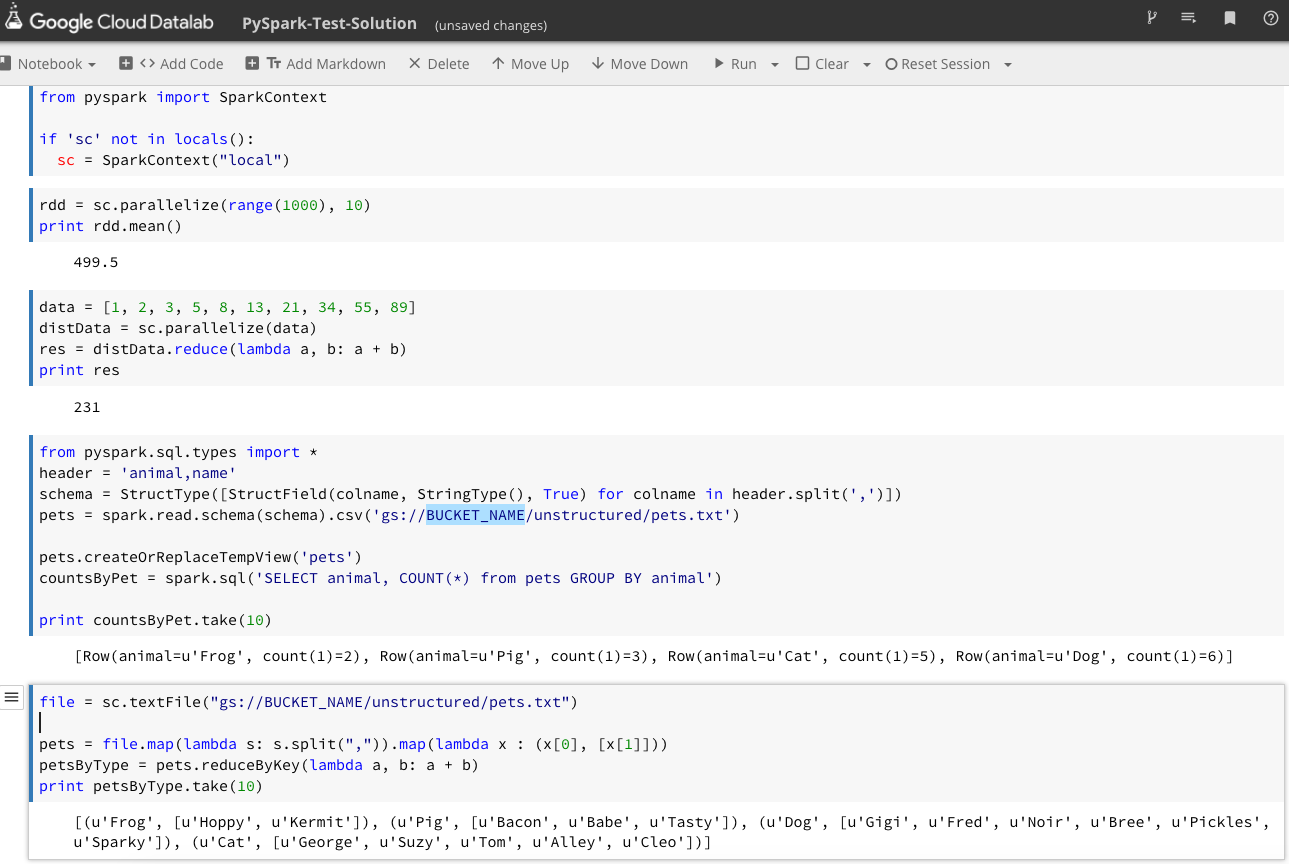
https://github.com/GoogleCloudPlatform/training-data-analyst



Then click on **Clone repository**

3. Back on the Datalab home page click the **Home** icon and navigate to datalab/notebooks/training-data-analyst/courses/unstructured. Click on **PySpark-Test-Solution.ipynb** to open that notebook.

4. In the notebook, Click on **Clear | All Cells**. Now, execute each cell in turn, making sure to change any occurrences of **BUCKET\_NAME** to be the name of your bucket.



5. You will want to stop this notebook as you did the previous one. Click the **Running Sessions** link on the right side of the toolbar. Then, click the **Shutdown** button to the left of the PySpark-Test-Solution notebook.

Close this tab and return to the Datalab home page.

**Add Machine Learning**

## Enabling Machine Learning APIs

1. Back on the Datalab home page click the **Home** icon and navigate to datalab/notebooks/training-data-analyst/courses/unstructured. Click on **ML-Tests-Solution.ipynb** to open that notebook.

2. Click **Clear | Clear All Cells**. In the first code block, notice that you need to generate an API key to use the machine learning APIs. In the **Google Cloud Platform Web Console**, from the Products and Services menu, choose **API Manager**. In the navigation pane on the left select **Credentials**. Then, click the **Create credentials** button and select **API key**. Copy the generated key to your clipboard and then click the **Close** button.

3. Go back to your Datalab notebook and paste the key you just generated over the text **ENTER API KEY HERE**. Also, enter you Google Cloud Platform project's ID over the text **ENTER-PROJECT-ID-HERE.** Now, click the **Run** button button to run that code block.

4. Go back to the Web Console and the API Manager page. Click on the **Library** link in the navigation pane. In the Search box type **Speech**. Then, click on the link to the **Google Cloud Speech API**. Click the **Enable**button on the resulting screen (if not already enabled).

5. Go back to the Library page and type **Translate** in the search page. As you did with the Speech API, enable the **Translation API** (if not already enabled). Repeat this process enabling the **Vision** and **Language** APIs.

6. Go back to your Datalab notebook and plugin the appropriate PROJECT and BUCKET names. Then, read the narrative and execute each cell in term.

## Testing Large Data Sets

1. Find the block of code that reads in Alice in Wonderland. Change the filename to **alice-in-wonderland-transformed.txt**. This will read the entire book.

2. In the BigQuery query, change the limit from 10 to 1000.

3. Click the drop-down next to the Run button and select **Run all cells**.

4. It will take a little while to run. Examine the results and the code. Experiment with the code if you like.

5. Click on the **Notebook** menu and select **Save and Checkpoint**. Then you can close the tab with this notebook.