Assignment 4

Name – Sati, Ankit Date – 3/17/2022

Section – 001

SID – as14128

Total in points (Maximum 100 points)–

Professors Comments –

Affirmation of Independent Effort – Ankit Sati

Index

Note to Grader - We have discussed with the professor that we only have to complete **4 of the 7 topics** in the assignment.

However I have done 6 and have only missed part 4.

Have discussed the same with professor as **HDInsight is a paid service.**

Part I - Completed

Part II - Completed

Part V - Completed

Part III - Completed

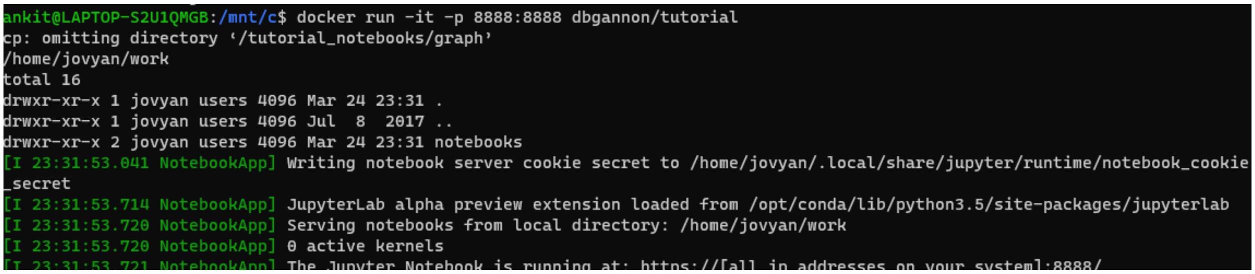
Part VI - Completed

Part VII - Completed

Part VII - HDinsight is a paid service – left this one. (Explained above)

Part I – Get Familiar with spark.

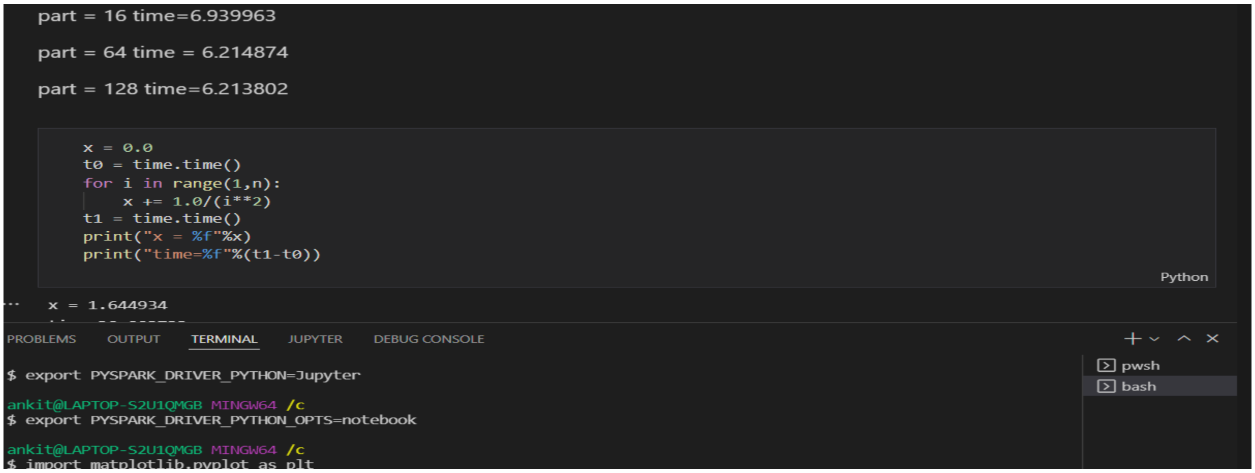
1. **Setting up resources to run the Spark Euler.**



```
ankit@LAPTOP-S2U1QMGB:/mnt/c$ docker run -it -p 8888:8888 dbgannon/tutorial
cp: omitting directory ‘/tutorial_notebooks/graph’
/home/jovyan/work
total 16
drwxr-xr-x 1 jovyan users 4096 Mar 24 23:31 .
drwxr-xr-x 1 jovyan users 4096 Jul  8  2017 ..
drwxr-xr-x 2 jovyan users 4096 Mar 24 23:31 notebooks
[I 23:31:53.041 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie
_secret
[I 23:31:53.714 NotebookApp] JupyterLab alpha preview extension loaded from /opt/conda/lib/python3.5/site-packages/jupyterlab
[I 23:31:53.720 NotebookApp] Serving notebooks from local directory: /home/jovyan/work
[I 23:31:53.720 NotebookApp] 0 active kernels
[I 23:31:53.721 NotebookApp] The Jupyter Notebook is running at: https://[all ip addresses on your system]:8888/
```

2. **Setting up the Eular-Spark in the same directory.**

- **$ export PYSPARK_DRIVER_PYTHON=Jupyter**
- **$ export PYSPARK_DRIVER_PYTHON_OPTS=notebook**

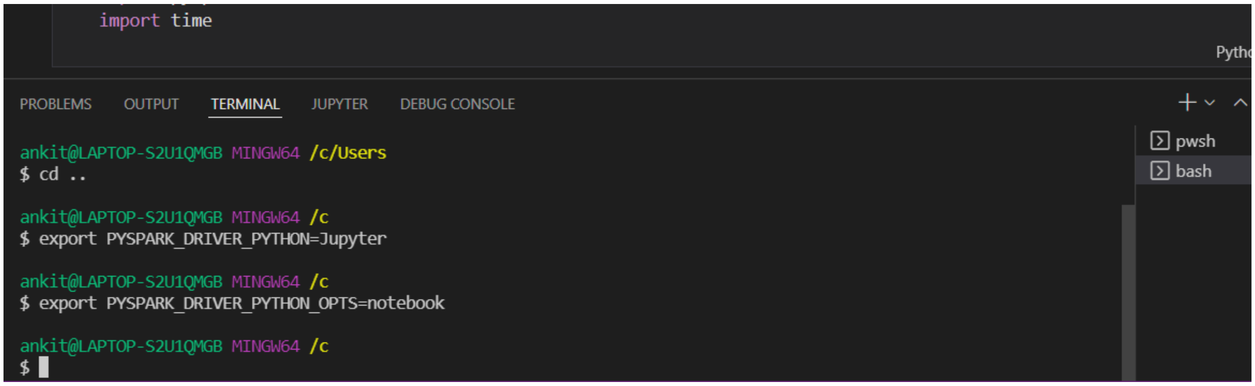


```
part = 16 time=6.939963

part = 64 time = 6.214874

part = 128 time=6.213802
```

```
x = 0.0
t0 = time.time()
for i in range(1,n):
    x += 1.0/(i**2)
t1 = time.time()
print("x = %f"%x)
print("time=%f"%(t1-t0))
```

```
x = 1.644934
```

```
$ export PYSPARK_DRIVER_PYTHON=Jupyter

ankit@LAPTOP-S2U1QMGB MINGW64 /c
$ export PYSPARK_DRIVER_PYTHON_OPTS=notebook

ankit@LAPTOP-S2U1QMGB MINGW64 /c
$ import matplotlib.pyplot as plt
```

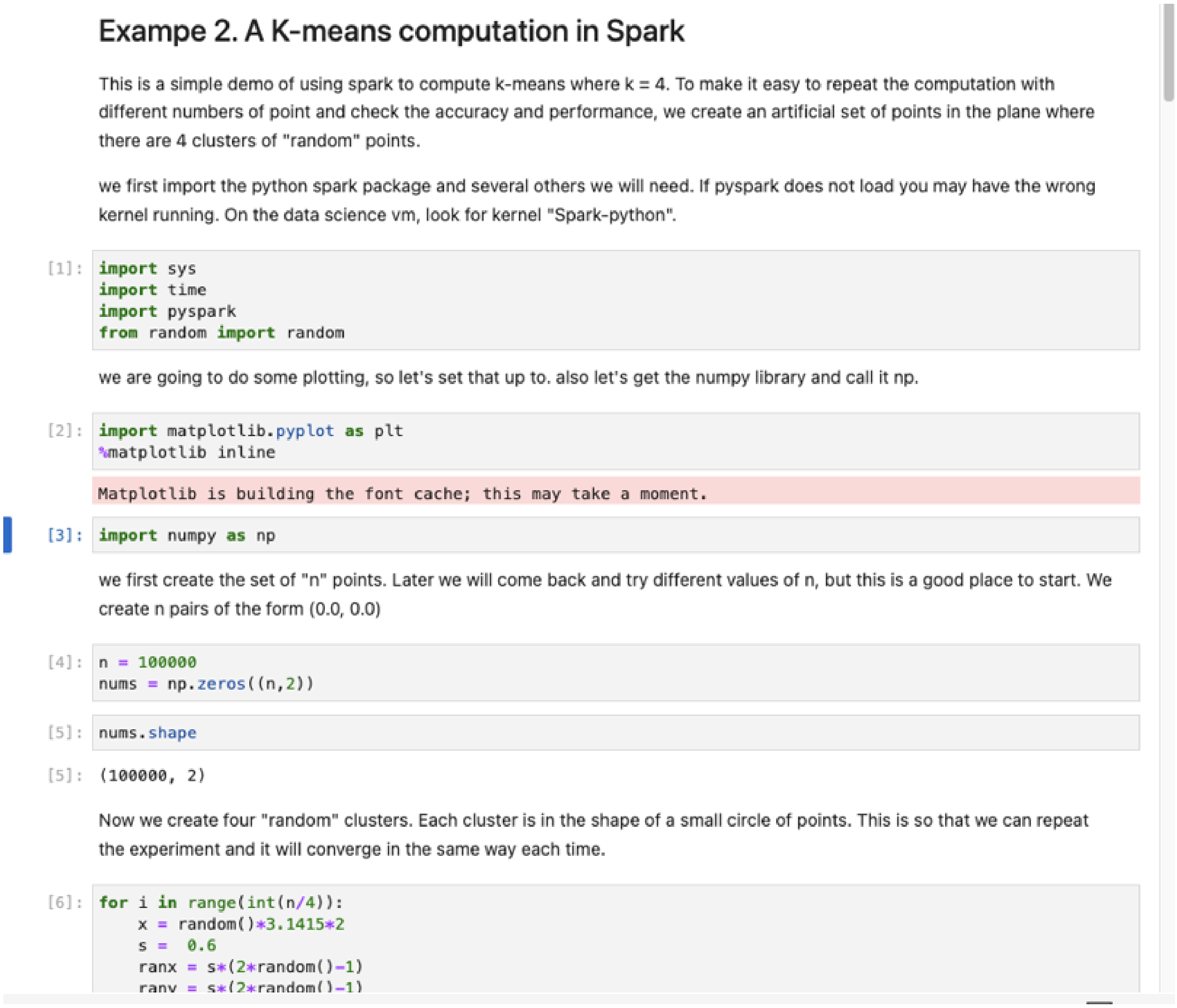


```
import time
```

```
ankit@LAPTOP-S2U1QMGB MINGW64 /c/Users
$ cd ..

ankit@LAPTOP-S2U1QMGB MINGW64 /c
$ export PYSPARK_DRIVER_PYTHON=Jupyter

ankit@LAPTOP-S2U1QMGB MINGW64 /c
$ export PYSPARK_DRIVER_PYTHON_OPTS=notebook

ankit@LAPTOP-S2U1QMGB MINGW64 /c
$
```

Second Notebook just to experiment with. I choose SQL-Magic

- pip install pandas
- pip install sqlalchemy # ORM for databases



Exampe 2. A K-means computation in Spark

This is a simple demo of using spark to compute k-means where k = 4. To make it easy to repeat the computation with different numbers of point and check the accuracy and performance, we create an artificial set of points in the plane where there are 4 clusters of "random" points.

we first import the python spark package and several others we will need. If pyspark does not load you may have the wrong kernel running. On the data science vm, look for kernel "Spark-python".

```
[1]: import sys
     import time
     import pyspark
     from random import random
```

we are going to do some plotting, so let's set that up to. also let's get the numpy library and call it np.

```
[2]: import matplotlib.pyplot as plt
     %matplotlib inline

     Matplotlib is building the font cache; this may take a moment.
```

```
[3]: import numpy as np
```

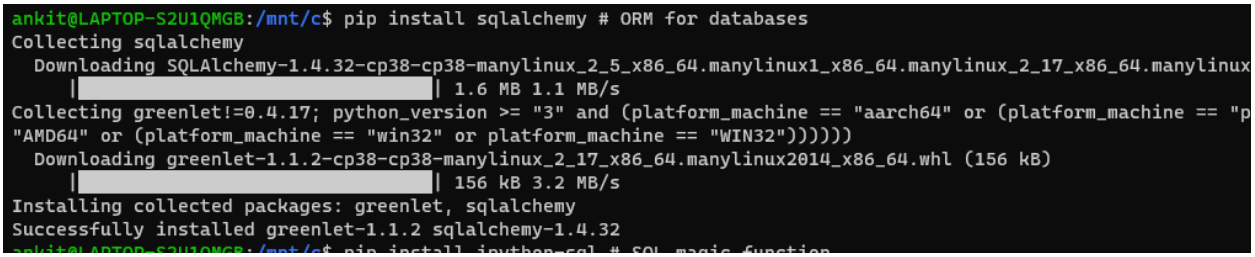
we first create the set of "n" points. Later we will come back and try different values of n, but this is a good place to start. We create n pairs of the form (0.0, 0.0)

```
[4]: n = 100000
     nums = np.zeros((n,2))
```

```
[5]: nums.shape

[5]: (100000, 2)
```

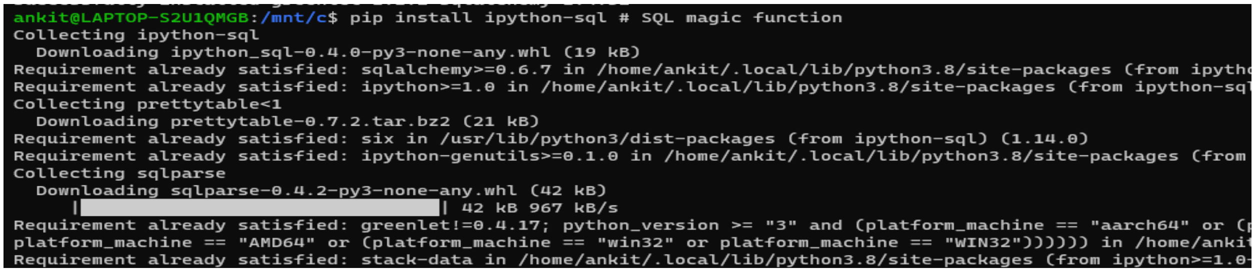
Now we create four "random" clusters. Each cluster is in the shape of a small circle of points. This is so that we can repeat the experiment and it will converge in the same way each time.

```
[6]: for i in range(int(n/4)):
         x = random()*3.1415*2
         s =  0.6
         ranx = s*(2*random()-1)
         rany = s*(2*random()-1)
```

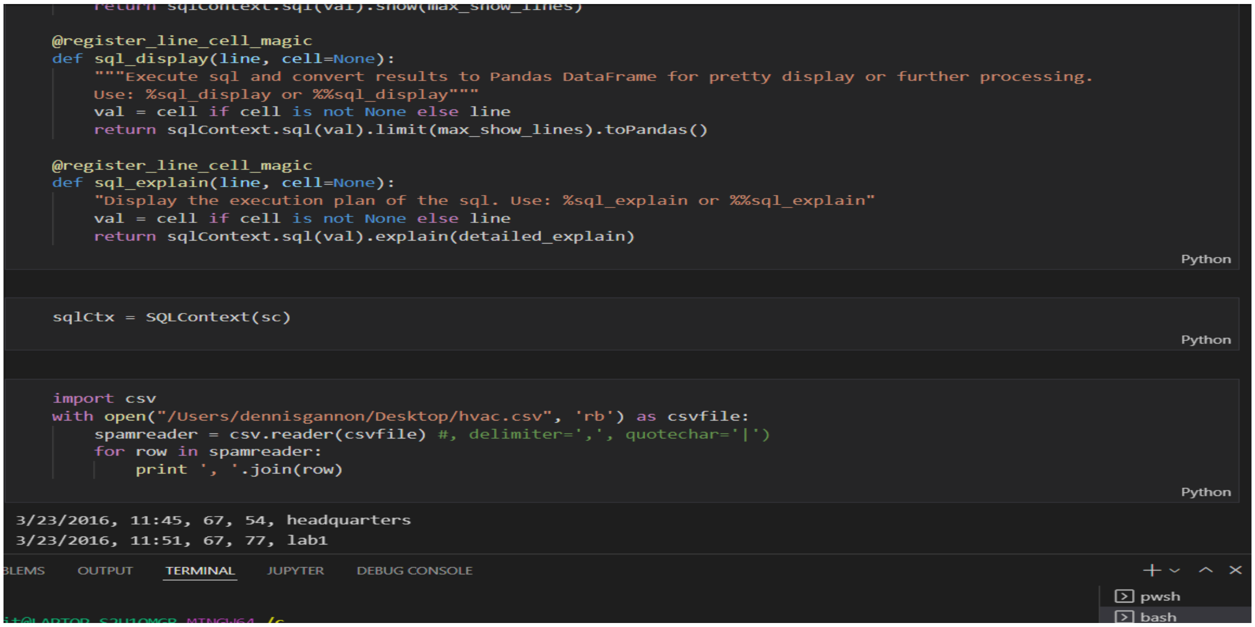


```
ankit@LAPTOP-S2U1QMGB:/mnt/c$ pip install sqlalchemy # ORM for databases
Collecting sqlalchemy
  Downloading SQLAlchemy-1.4.32-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
     |████████████████████████████████| 1.6 MB 1.1 MB/s
Collecting greenlet!=0.4.17; python_version >= "3" and (platform_machine == "aarch64" or (platform_machine == "p
"AMD64" or (platform_machine == "win32" or platform_machine == "WIN32"))))))
  Downloading greenlet-1.1.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (156 kB)
     |████████████████████████████████| 156 kB 3.2 MB/s
Installing collected packages: greenlet, sqlalchemy
Successfully installed greenlet-1.1.2 sqlalchemy-1.4.32
ankit@LAPTOP-S2U1QMGB:/mnt/c$ pip install ipython-sql # SQL magic function
```

- pip install ipython-sql # SQL magic function



```
ankit@LAPTOP-S2U1QMGB:/mnt/c$ pip install ipython-sql # SQL magic function
Collecting ipython-sql
  Downloading ipython_sql-0.4.0-py3-none-any.whl (19 kB)
Requirement already satisfied: sqlalchemy>=0.6.7 in /home/ankit/.local/lib/python3.8/site-packages (from ipytho
Requirement already satisfied: ipython>=1.0 in /home/ankit/.local/lib/python3.8/site-packages (from ipython-sql
Collecting prettytable<1
  Downloading prettytable-0.7.2.tar.bz2 (21 kB)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from ipython-sql) (1.14.0)
Requirement already satisfied: ipython-genutils>=0.1.0 in /home/ankit/.local/lib/python3.8/site-packages (from
Collecting sqlparse
  Downloading sqlparse-0.4.2-py3-none-any.whl (42 kB)
     |████████████████████████████████| 42 kB 967 kB/s
Requirement already satisfied: greenlet!=0.4.17; python_version >= "3" and (platform_machine == "aarch64" or (p
platform_machine == "AMD64" or (platform_machine == "win32" or platform_machine == "WIN32")))))) in /home/ankit
Requirement already satisfied: stack-data in /home/ankit/.local/lib/python3.8/site-packages (from ipython>=1.0-
```



```
        return sqlContext.sql(val).show(max_show_lines)

    @register_line_cell_magic
    def sql_display(line, cell=None):
        """Execute sql and convert results to Pandas DataFrame for pretty display or further processing.
        Use: %sql_display or %%sql_display"""
        val = cell if cell is not None else line
        return sqlContext.sql(val).limit(max_show_lines).toPandas()

    @register_line_cell_magic
    def sql_explain(line, cell=None):
        "Display the execution plan of the sql. Use: %sql_explain or %%sql_explain"
        val = cell if cell is not None else line
        return sqlContext.sql(val).explain(detailed_explain)
```

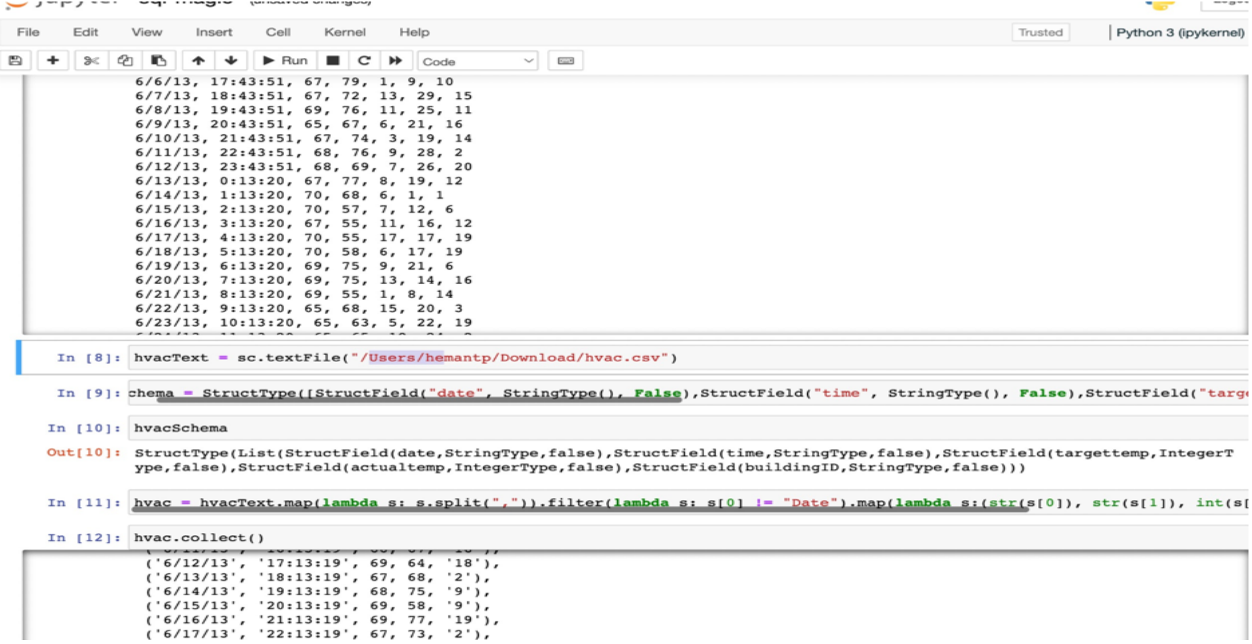
```
    sqlCtx = SQLContext(sc)
```

```
    import csv
    with open("/Users/dennisgannon/Desktop/hvac.csv", 'rb') as csvfile:
        spamreader = csv.reader(csvfile) #, delimiter=',', quotechar='|')
        for row in spamreader:
            print ', '.join(row)
```

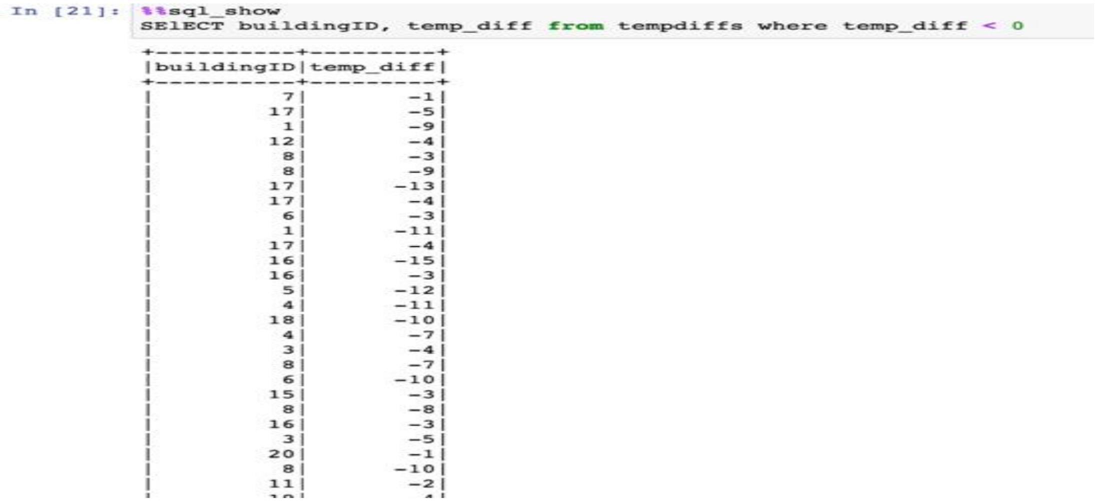
```
3/23/2016, 11:45, 67, 54, headquarters
3/23/2016, 11:51, 67, 77, lab1
```

3. Sql-magic

Finally now we can go ahead and setup the SQL magic over the Jupyter Notebook.
We will be using the same notebook as shared in the example in the tutorial.



```
        6/6/13, 17:43:51, 67, 79, 1, 9, 10
        6/7/13, 18:43:51, 67, 72, 13, 29, 15
        6/8/13, 19:43:51, 69, 76, 11, 25, 11
        6/9/13, 20:43:51, 65, 67, 6, 21, 16
        6/10/13, 21:43:51, 67, 74, 3, 19, 14
        6/11/13, 22:43:51, 68, 76, 9, 28, 2
        6/12/13, 23:43:51, 68, 69, 7, 26, 20
        6/13/13, 0:13:20, 67, 77, 8, 19, 12
        6/14/13, 1:13:20, 70, 68, 6, 1, 1
        6/15/13, 2:13:20, 70, 57, 7, 12, 6
        6/16/13, 3:13:20, 67, 55, 11, 16, 12
        6/17/13, 4:13:20, 70, 55, 17, 17, 19
        6/18/13, 5:13:20, 70, 58, 6, 17, 19
        6/19/13, 6:13:20, 69, 75, 9, 21, 6
        6/20/13, 7:13:20, 69, 75, 13, 14, 16
        6/21/13, 8:13:20, 69, 55, 1, 8, 14
        6/22/13, 9:13:20, 65, 68, 15, 20, 3
        6/23/13, 10:13:20, 65, 63, 5, 22, 19

 In [8]: hvacText = sc.textFile("/Users/hemantp/Download/hvac.csv")

 In [9]: chema = StructType([StructField("date", StringType(), False),StructField("time", StringType(), False),StructField("targ

In [10]: hvacSchema

Out[10]: StructType(List(StructField(date,StringType,false),StructField(time,StringType,false),StructField(targettemp,IntegerT
         ype,false),StructField(actualtemp,IntegerType,false),StructField(buildingID,StringType,false)))

In [11]: hvac = hvacText.map(lambda s: s.split(",")).filter(lambda s: s[0] != "Date").map(lambda s:(str(s[0]), str(s[1]), int(s[

In [12]: hvac.collect()

          ('6/12/13', '17:13:19', 69, 64, '18'),
          ('6/13/13', '18:13:19', 67, 68, '2'),
          ('6/14/13', '19:13:19', 68, 75, '9'),
          ('6/15/13', '20:13:19', 69, 58, '9'),
          ('6/16/13', '21:13:19', 69, 77, '19'),
          ('6/17/13', '22:13:19', 67, 73, '2'),
```



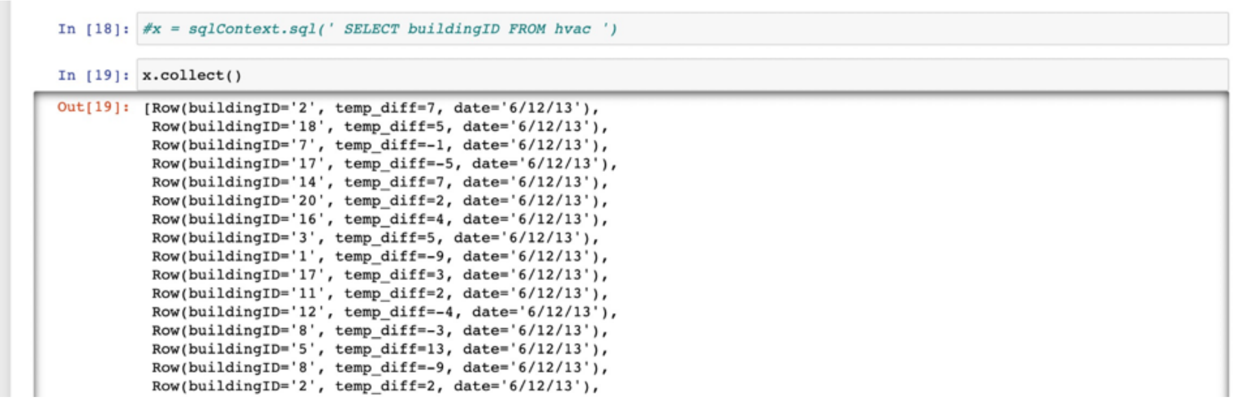
```
In [21]: %%sql_show
         SElECT buildingID, temp_diff from tempdiffs where temp_diff < 0

         +----------+---------+
         |buildingID|temp_diff|
         +----------+---------+
         |         7|       -1|
         |        17|       -5|
         |         1|       -9|
         |        12|       -4|
         |         8|       -3|
         |         8|       -9|
         |        17|      -13|
         |        17|       -4|
         |         6|       -3|
         |         1|      -11|
         |        17|       -4|
         |        16|      -15|
         |        16|       -3|
         |         5|      -12|
         |         4|      -11|
         |        18|      -10|
         |         4|       -7|
         |         3|       -4|
         |         8|       -7|
         |         6|      -10|
         |        15|       -3|
         |         8|       -8|
         |        16|       -3|
         |         3|       -5|
         |        20|       -1|
         |         8|      -10|
         |        11|       -2|
```

Now finally we need to check the build of this data on the SQL notebook, for that we will just collect the data provided in the first shard.

Total number of records

- We will have have a total of 50 records in the entire file

Command – sqlContect.sql(‘A,B,C’)



```
In [18]: #x = sqlContext.sql(' SELECT buildingID FROM hvac ')

In [19]: x.collect()

Out[19]: [Row(buildingID='2', temp_diff=7, date='6/12/13'),
          Row(buildingID='18', temp_diff=5, date='6/12/13'),
          Row(buildingID='7', temp_diff=-1, date='6/12/13'),
          Row(buildingID='17', temp_diff=-5, date='6/12/13'),
          Row(buildingID='14', temp_diff=7, date='6/12/13'),
          Row(buildingID='20', temp_diff=2, date='6/12/13'),
          Row(buildingID='16', temp_diff=4, date='6/12/13'),
          Row(buildingID='3', temp_diff=5, date='6/12/13'),
          Row(buildingID='1', temp_diff=-9, date='6/12/13'),
          Row(buildingID='17', temp_diff=3, date='6/12/13'),
          Row(buildingID='11', temp_diff=2, date='6/12/13'),
          Row(buildingID='12', temp_diff=-4, date='6/12/13'),
          Row(buildingID='8', temp_diff=-3, date='6/12/13'),
          Row(buildingID='5', temp_diff=13, date='6/12/13'),
          Row(buildingID='8', temp_diff=-9, date='6/12/13'),
          Row(buildingID='2', temp_diff=2, date='6/12/13'),
```

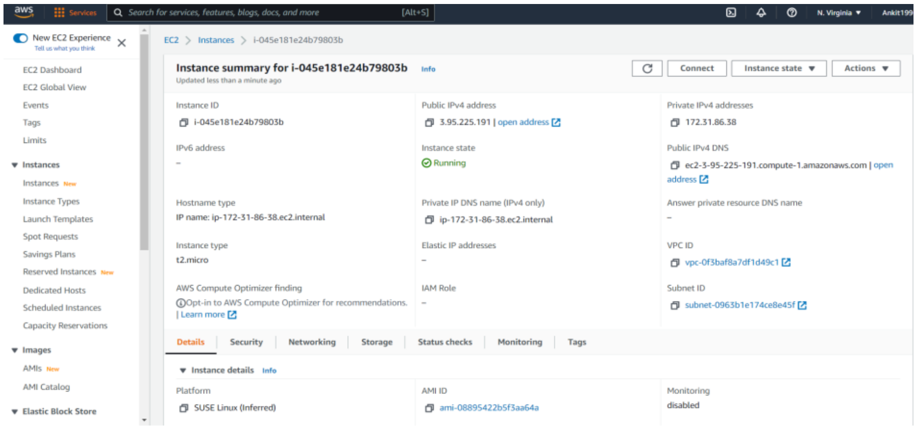
Part II: Experiment with Spark on AWS.

Amazon Elastic Map Reduce (EMR)

➔ EC2(**Compute server**) – This is basically a **regular server** instance that is used to deploy and the required resources over the VM as per the choices made by the users.
- This is used to deploy the VM.
- Manage resources over those VM's.
- Finally to migrate services and monitor volumes.

➔ S3(**Storage utility**) – This is a basic protocol that acts like a storage bucket.
- The prime feature of this protocol is to deal with the data as per service request.
- We need this to store the data in the **data buckets** which are later used to store and move the data across volumes created.

No we need to setup the cluster and run a stream to analyze on top of it.



EC2 > Instances > i-045e181e24b79803b

Instance summary for i-045e181e24b79803b Info

Updated less than a minute ago

Instance ID: i-045e181e24b79803b

Public IPv4 address: 3.95.225.191 | open address

Private IPv4 addresses: 172.31.86.38

IPv6 address: –

Instance state: Running

Public IPv4 DNS: ec2-3-95-225-191.compute-1.amazonaws.com | open address

Hostname type: IP name: ip-172-31-86-38.ec2.internal

Private IP DNS name (IPv4 only): ip-172-31-86-38.ec2.internal

Answer private resource DNS name: –

Instance type: t2.micro

Elastic IP addresses: –

VPC ID: vpc-0f3baf8a7df1d49c1

AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IAM Role: –

Subnet ID: subnet-0963b1e174ce8e45f

▼ Instance details Info

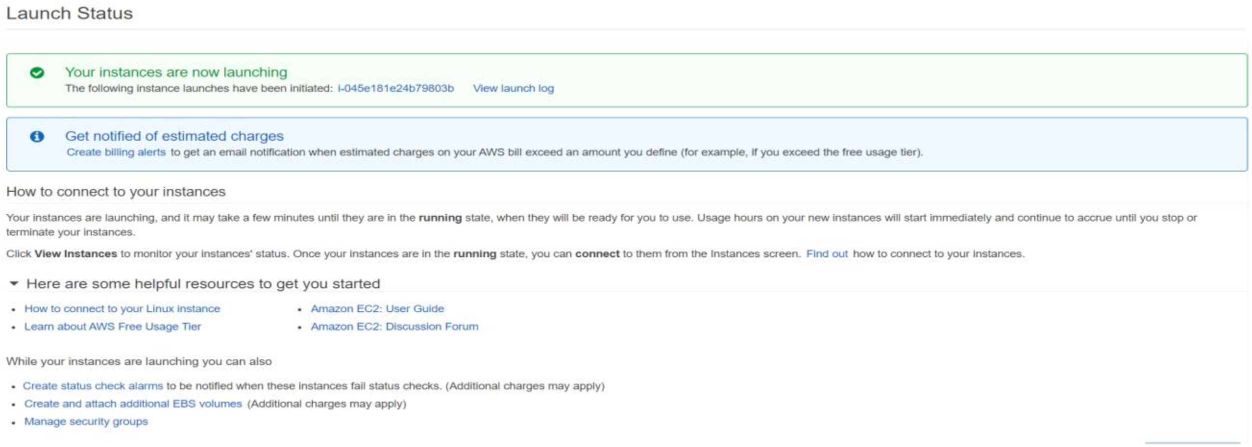
Platform: SUSE Linux (Inferred)

AMI ID: ami-08895422b5f3aa64a

Monitoring: disabled

Experimenting with Elastic Map Reduce on the minimal cluster.

Step 2 –



Launch Status

Your instances are now launching

The following instance launches have been initiated: i-045e181e24b79803b   View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

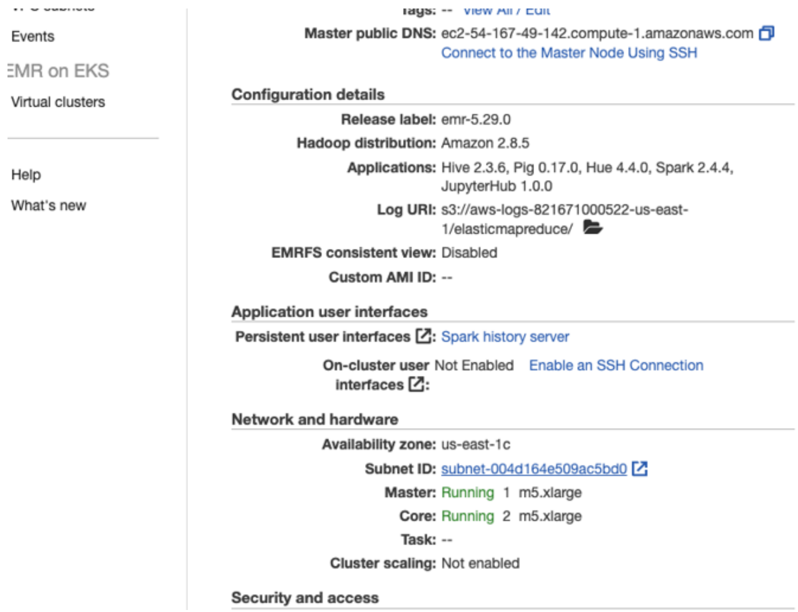
▼ Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

- Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
- Create and attach additional EBS volumes (Additional charges may apply)
- Manage security groups

Amazon Cluster Info



Tags: -- View All / Edit
Master public DNS: ec2-54-167-49-142.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

Release label: emr-5.29.0
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0, Spark 2.4.4, JupyterHub 1.0.0
Log URI: s3://aws-logs-821671000522-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

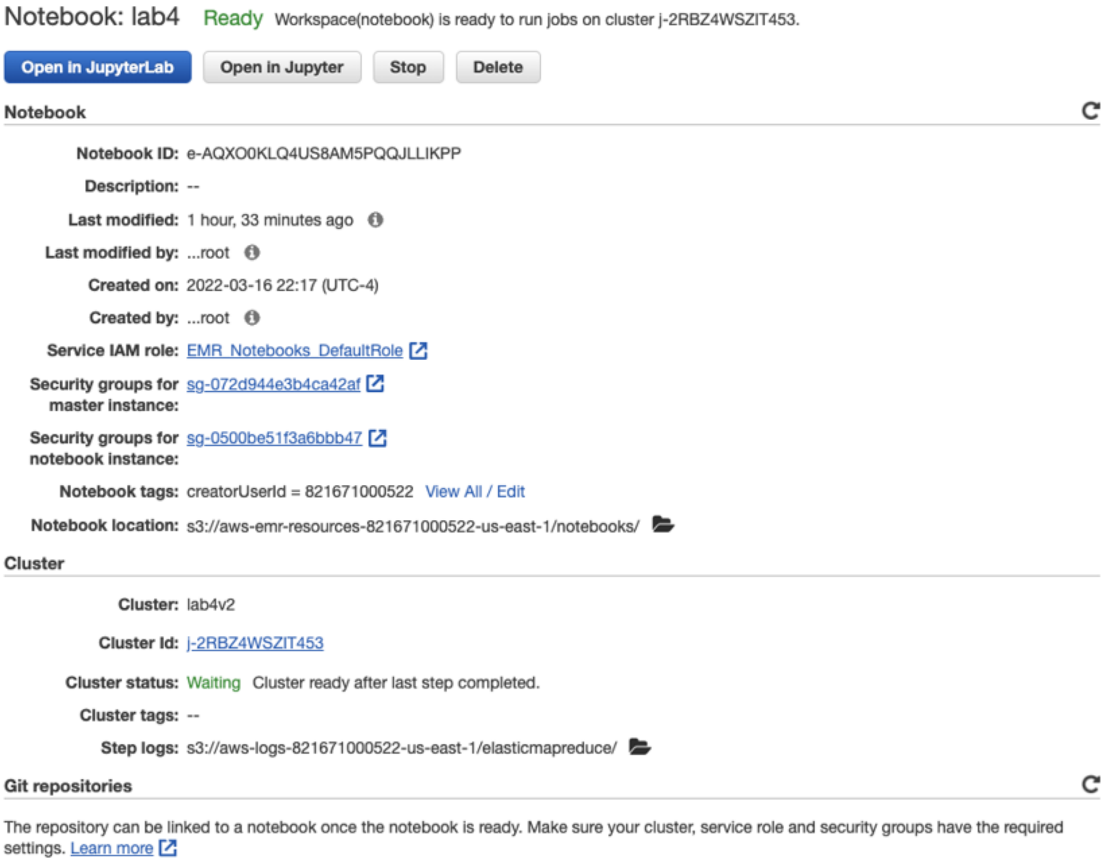
Persistent user interfaces: Spark history server
On-cluster user interfaces: Not Enabled Enable an SSH Connection

Network and hardware

Availability zone: us-east-1c
Subnet ID: subnet-004d164e509ac5bd0
Master: Running 1 m5.xlarge
Core: Running 2 m5.xlarge
Task: --
Cluster scaling: Not enabled

Security and access

We need to make sure that the stream is ready.



Notebook: lab4 Ready Workspace(notebook) is ready to run jobs on cluster j-2RBZ4WSZIT453.

Notebook

Notebook ID: e-AQXO0KLQ4US8AM5PQQJLLIKPP
Description: --
Last modified: 1 hour, 33 minutes ago
Last modified by: ...root
Created on: 2022-03-16 22:17 (UTC-4)
Created by: ...root
Service IAM role: EMR_Notebooks_DefaultRole
Security groups for master instance: sg-072d944e3b4ca42af
Security groups for notebook instance: sg-0500be51f3a6bbb47
Notebook tags: creatorUserId = 821671000522 View All / Edit
Notebook location: s3://aws-emr-resources-821671000522-us-east-1/notebooks/

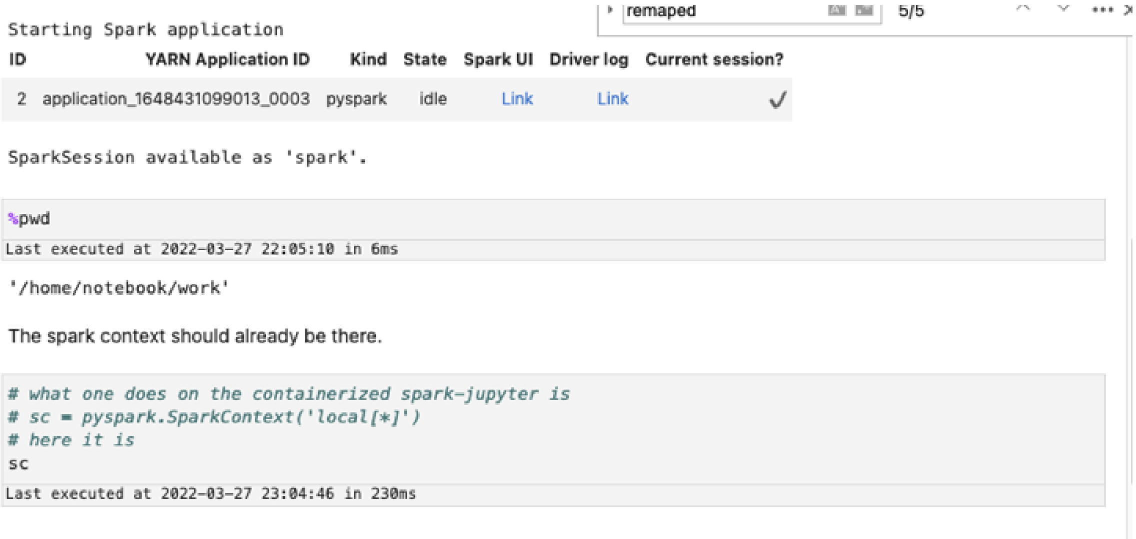
Cluster

Cluster: lab4v2
Cluster Id: j-2RBZ4WSZIT453
Cluster status: Waiting Cluster ready after last step completed.
Cluster tags: --
Step logs: s3://aws-logs-821671000522-us-east-1/elasticmapreduce/

Git repositories

The repository can be linked to a notebook once the notebook is ready. Make sure your cluster, service role and security groups have the required settings. Learn more

Next we need to begun the EMR and work with the yarn data in livy2.
IP address - **“hdfs://ip-172-31-22-51.ec2.internal:8020/user/wiki/text.txt”**
Text file = **txtfile.repartion(10)**
Array - Finally we need to put these values onto the array in ses2



```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
2	application_1648431099013_0003	pyspark	idle	Link	Link	✓

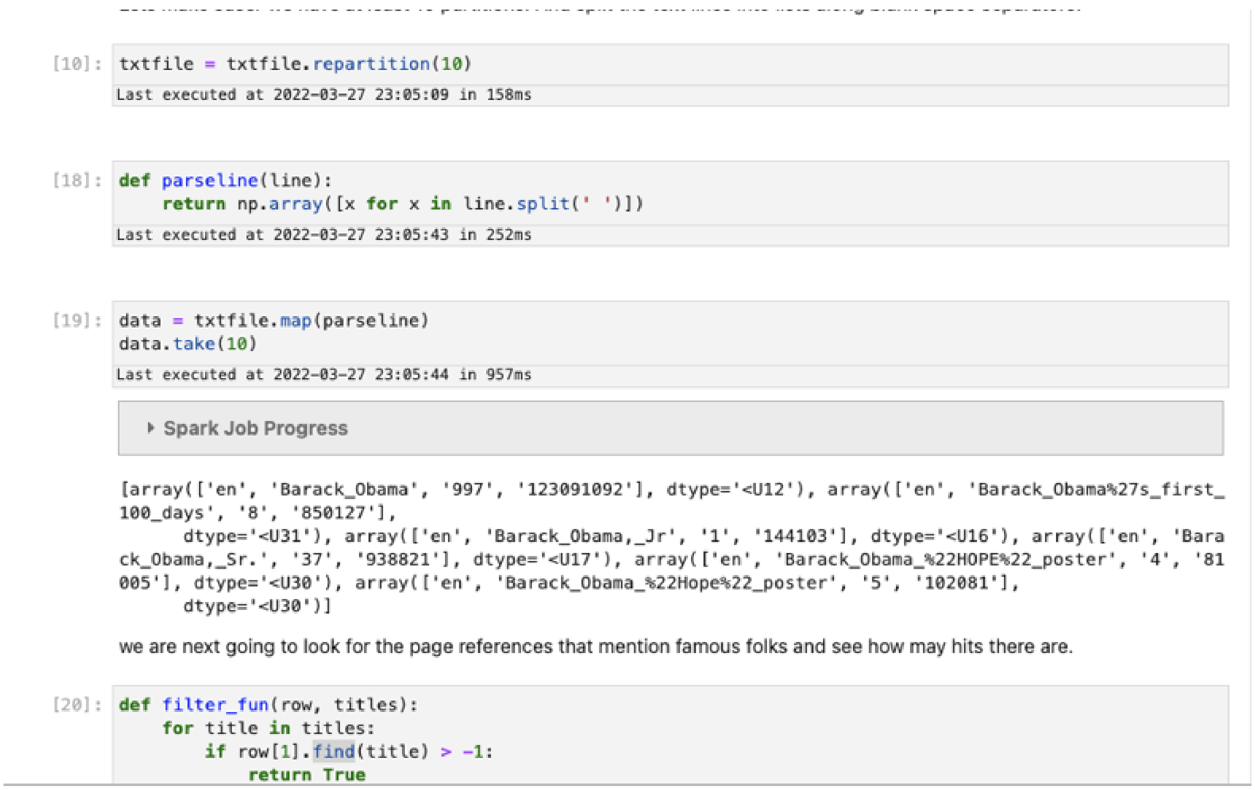
```
SparkSession available as 'spark'.
```

```
[2]: %pwd
Last executed at 2022-03-27 22:05:10 in 6ms
[2]: '/home/notebook/work'
```

The spark context should already be there.

```
[2]: # what one does on the containerized spark-jupyter is
     # sc = pyspark.SparkContext('local[*]')
     # here it is
     sc
Last executed at 2022-03-27 23:04:46 in 230ms
```

Text file = **txtfile.repartion(10)**
Splitting this test lines into black spaces.



```
[10]: txtfile = txtfile.repartition(10)
Last executed at 2022-03-27 23:05:09 in 158ms
```

```
[18]: def parseline(line):
          return np.array([x for x in line.split(' ')])
Last executed at 2022-03-27 23:05:43 in 252ms
```

```
[19]: data = txtfile.map(parseline)
      data.take(10)
Last executed at 2022-03-27 23:05:44 in 957ms
```

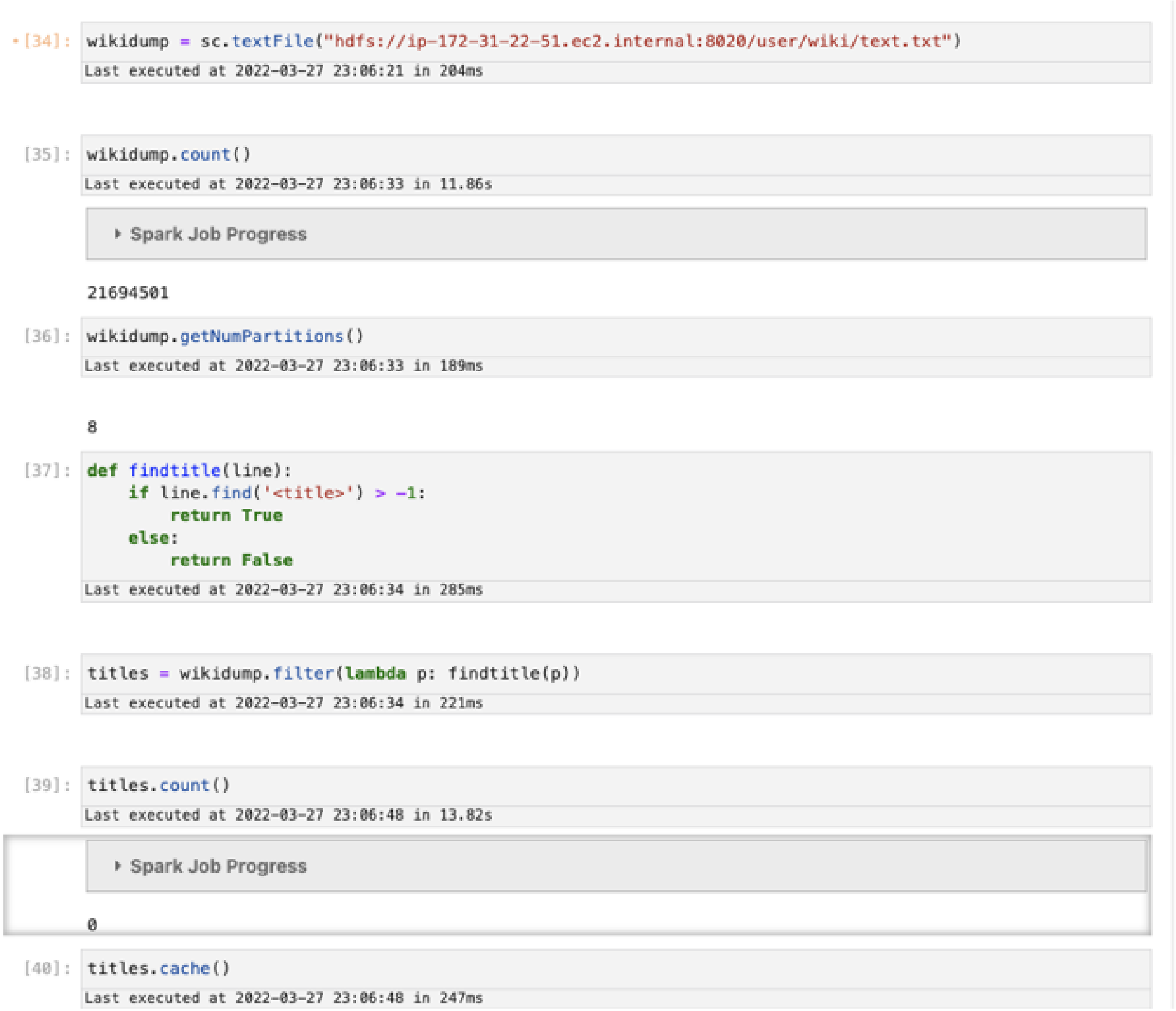
▸ Spark Job Progress

```
[array(['en', 'Barack_Obama', '997', '123091092'], dtype='<U12'), array(['en', 'Barack_Obama%27s_first_
100_days', '8', '850127'],
      dtype='<U31'), array(['en', 'Barack_Obama,_Jr', '1', '144103'], dtype='<U16'), array(['en', 'Bara
ck_Obama,_Sr.', '37', '938821'], dtype='<U17'), array(['en', 'Barack_Obama_%22HOPE%22_poster', '4', '81
005'], dtype='<U30'), array(['en', 'Barack_Obama_%22Hope%22_poster', '5', '102081'],
      dtype='<U30')]
```

we are next going to look for the page references that mention famous folks and see how may hits there are.

```
[20]: def filter_fun(row, titles):
          for title in titles:
              if row[1].find(title) > -1:
                  return True
```

Now finally we will complete the experiment by extracting the heading from each and every line.

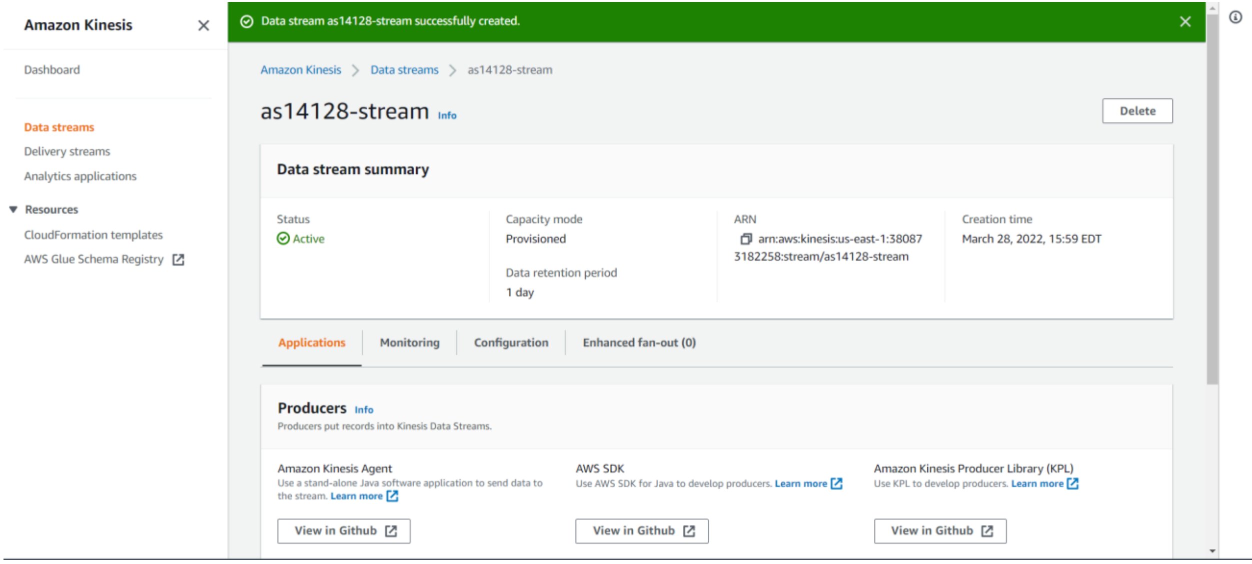


```
•[34]: wikidump = sc.textFile("hdfs://ip-172-31-22-51.ec2.internal:8020/user/wiki/text.txt")
       Last executed at 2022-03-27 23:06:21 in 204ms

 [35]: wikidump.count()
       Last executed at 2022-03-27 23:06:33 in 11.86s

        ▸ Spark Job Progress

       21694501

 [36]: wikidump.getNumPartitions()
       Last executed at 2022-03-27 23:06:33 in 189ms

       8

 [37]: def findtitle(line):
           if line.find('<title>') > -1:
               return True
           else:
               return False
       Last executed at 2022-03-27 23:06:34 in 285ms

 [38]: titles = wikidump.filter(lambda p: findtitle(p))
       Last executed at 2022-03-27 23:06:34 in 221ms

 [39]: titles.count()
       Last executed at 2022-03-27 23:06:48 in 13.82s

        ▸ Spark Job Progress

       0

 [40]: titles.cache()
       Last executed at 2022-03-27 23:06:48 in 247ms
```

Part V: Experiment with Streaming Big Data on AWS; execute and document the following Exercise.

Note – In this case I have used the same json repository mentioned in tutorial as if we take the other one it will shoot up the cost because we have chosen provisioned data. File chosen - .json available with tutorial.

1. Setup the kinesis account on amazon and make a new strem. Stream name – as14128-stream. This is just a dummy stream without any data as of this moment.



Data stream as14128-stream successfully created.

as14128-stream

Data stream summary

Status: Active

Capacity mode: Provisioned

Data retention period: 1 day

ARN: arn:aws:kinesis:us-east-1:380873182258:stream/as14128-stream

Creation time: March 28, 2022, 15:59 EDT

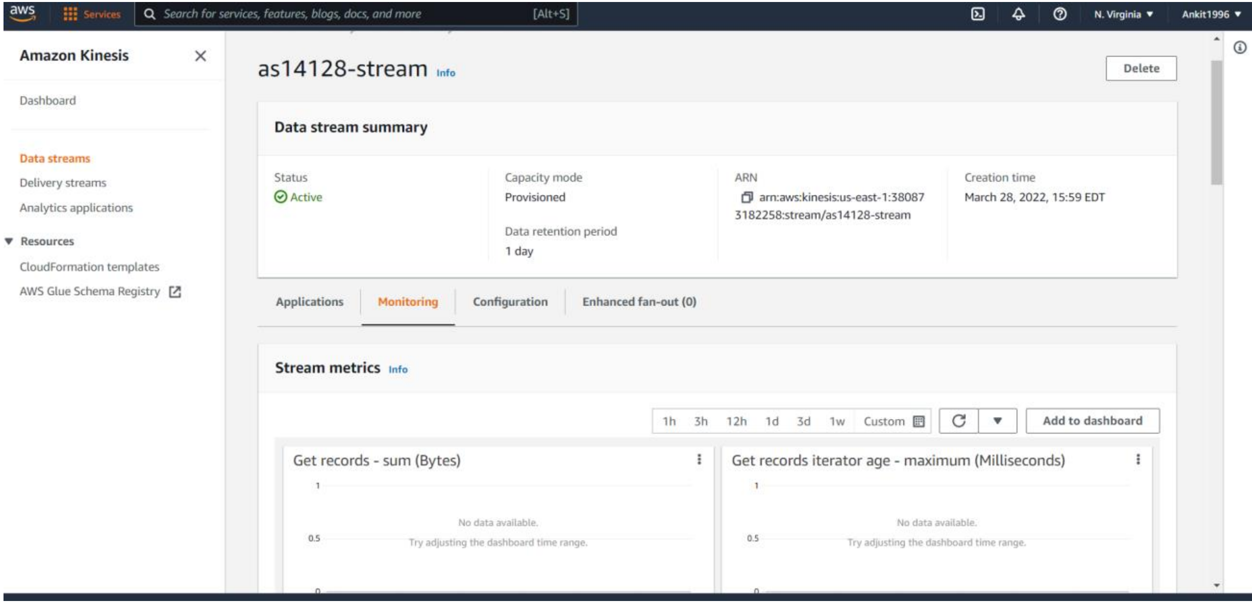
Producers: Producers put records into Kinesis Data Streams.

Amazon Kinesis Agent: Use a stand-alone Java software application to send data to the stream. Learn more

AWS SDK: Use AWS SDK for Java to develop producers. Learn more

Amazon Kinesis Producer Library (KPL): Use KPL to develop producers. Learn more

2. Next step is to check if all the things are in place and see that the monitoring **charts** are empty.



as14128-stream

Data stream summary

Status: Active

Capacity mode: Provisioned

Data retention period: 1 day

ARN: arn:aws:kinesis:us-east-1:380873182258:stream/as14128-stream

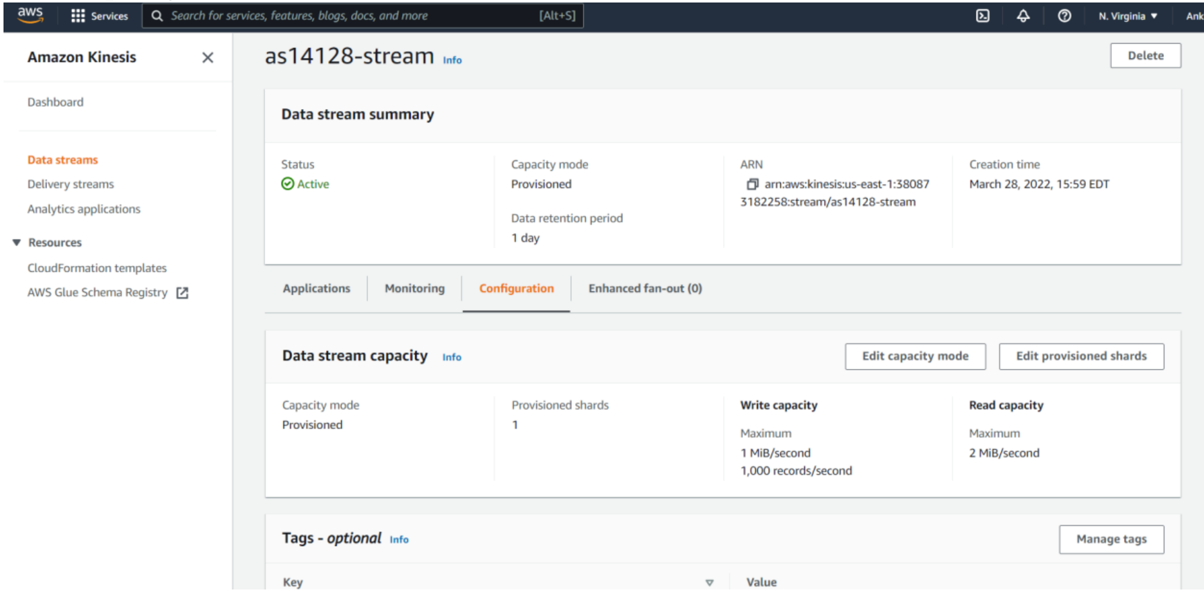
Creation time: March 28, 2022, 15:59 EDT

Stream metrics

Get records - sum (Bytes): No data available. Try adjusting the dashboard time range.

Get records iterator age - maximum (Milliseconds): No data available. Try adjusting the dashboard time range.

3. Now that we have checked that the charts are empty, we need to go ahead make sure that we have noted don the **configuration** for this stream as it will come in handy later in the process.
   - ➔ Capacity Mode – Provisioned
   - ➔ Total Shards -1 (Only passing a single file onto this stream so no point in getting more)
   - ➔ Write capacity on Stream – 1Mib/Second (The data will come in pretty quick so need to stop to reduce the cost)
   - ➔ Read capacity – 2 Mib/second



as14128-stream

Data stream summary

Status: Active

Capacity mode: Provisioned

Data retention period: 1 day

ARN: arn:aws:kinesis:us-east-1:380873182258:stream/as14128-stream

Creation time: March 28, 2022, 15:59 EDT

Data stream capacity

Capacity mode: Provisioned

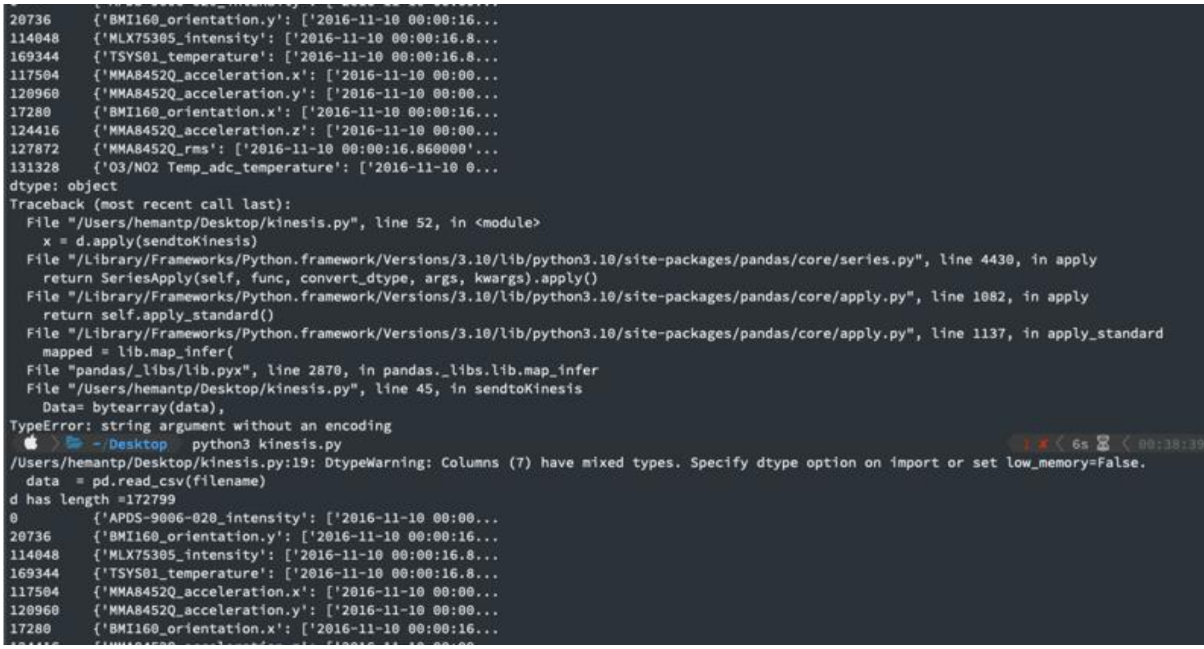
Provisioned shards: 1

Write capacity: Maximum 1 MiB/second, 1,000 records/second

Read capacity: Maximum 2 MiB/second

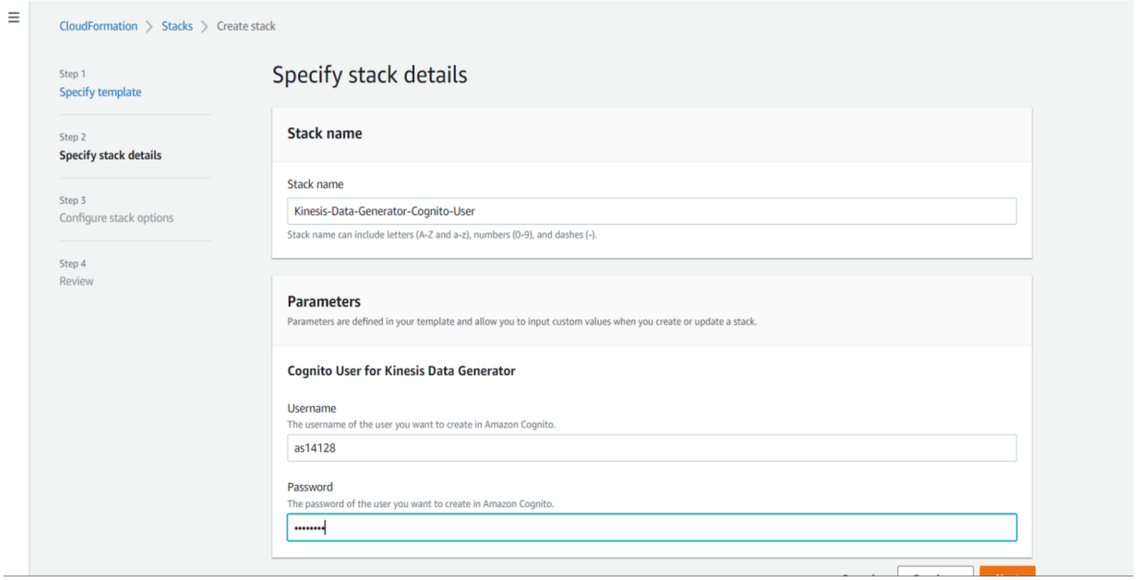
Tags - *optional*

Key	Value



```
20736     {'BMI160_orientation.y': ['2016-11-10 00:00:16...
114048    {'MLX75305_intensity': ['2016-11-10 00:00:16.8...
169344    {'TSYS01_temperature': ['2016-11-10 00:00:16.8...
117504    {'MMA8452Q_acceleration.x': ['2016-11-10 00:00...
120960    {'MMA8452Q_acceleration.y': ['2016-11-10 00:00...
17280     {'BMI160_orientation.x': ['2016-11-10 00:00:16...
124416    {'MMA8452Q_acceleration.z': ['2016-11-10 00:00...
127872    {'MMA8452Q_rms': ['2016-11-10 00:00:16.860000'...
131328    {'O3/NO2 Temp_adc_temperature': ['2016-11-10 0...
dtype: object
Traceback (most recent call last):
  File "/Users/hemantp/Desktop/kinesis.py", line 52, in <module>
    x = d.apply(sendtoKinesis)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/pandas/core/series.py", line 4430, in apply
    return SeriesApply(self, func, convert_dtype, args, kwargs).apply()
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/pandas/core/apply.py", line 1082, in apply
    return self.apply_standard()
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/pandas/core/apply.py", line 1137, in apply_standard
    mapped = lib.map_infer(
  File "pandas/_libs/lib.pyx", line 2870, in pandas._libs.lib.map_infer
  File "/Users/hemantp/Desktop/kinesis.py", line 45, in sendtoKinesis
    Data= bytearray(data),
TypeError: string argument without an encoding
  ~/Desktop  python3 kinesis.py                                                      6s  00:38:39
/Users/hemantp/Desktop/kinesis.py:19: DtypeWarning: Columns (7) have mixed types. Specify dtype option on import or set low_memory=False.
  data  = pd.read_csv(filename)
d has length =172799
0         {'APDS-9006-020_intensity': ['2016-11-10 00:00...
20736     {'BMI160_orientation.y': ['2016-11-10 00:00:16...
114048    {'MLX75305_intensity': ['2016-11-10 00:00:16.8...
169344    {'TSYS01_temperature': ['2016-11-10 00:00:16.8...
117504    {'MMA8452Q_acceleration.x': ['2016-11-10 00:00...
120960    {'MMA8452Q_acceleration.y': ['2016-11-10 00:00...
17280     {'BMI160_orientation.x': ['2016-11-10 00:00:16...
```

4. Now we need to create a new Stack on top op which we are going to overlay the data. In our case we are taking a small .json file so we will be using the basic stack.
   Creating account on KDG



Specify stack details

Stack name

Stack name: Kinesis-Data-Generator-Cognito-User

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

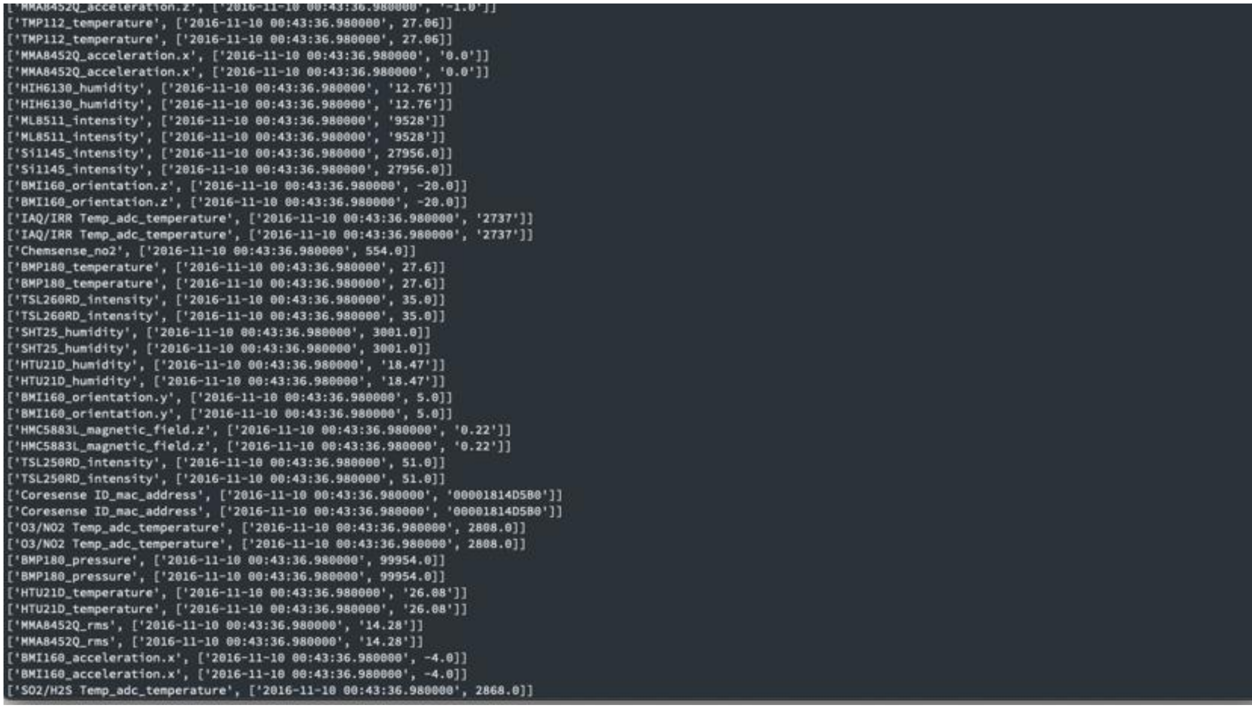
Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Cognito User for Kinesis Data Generator

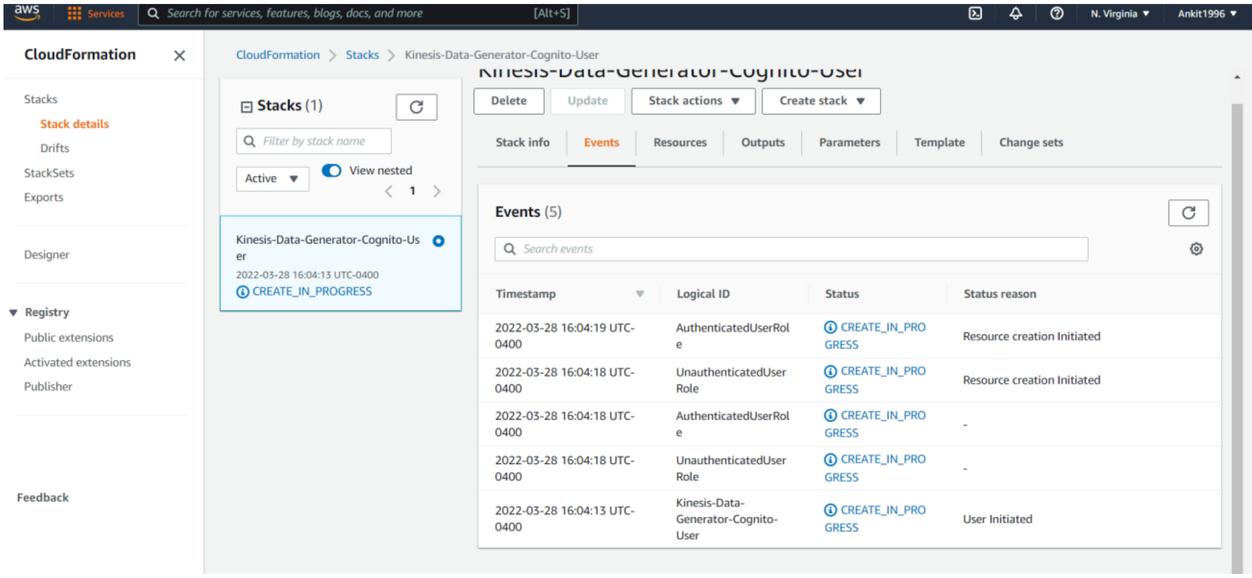
Username
The username of the user you want to create in Amazon Cognito.
as14128

Password
The password of the user you want to create in Amazon Cognito.
••••••••



```
['MMA8452Q_acceleration.z', ['2016-11-10 00:43:36.980000', '-1.0']]
['TMP112_temperature', ['2016-11-10 00:43:36.980000', 27.06]]
['TMP112_temperature', ['2016-11-10 00:43:36.980000', 27.06]]
['MMA8452Q_acceleration.x', ['2016-11-10 00:43:36.980000', '0.0']]
['MMA8452Q_acceleration.x', ['2016-11-10 00:43:36.980000', '0.0']]
['HIH6130_humidity', ['2016-11-10 00:43:36.980000', '12.76']]
['HIH6130_humidity', ['2016-11-10 00:43:36.980000', '12.76']]
['ML8511_intensity', ['2016-11-10 00:43:36.980000', '9528']]
['ML8511_intensity', ['2016-11-10 00:43:36.980000', '9528']]
['Si1145_intensity', ['2016-11-10 00:43:36.980000', 27956.0]]
['Si1145_intensity', ['2016-11-10 00:43:36.980000', 27956.0]]
['BMI160_orientation.z', ['2016-11-10 00:43:36.980000', -20.0]]
['BMI160_orientation.z', ['2016-11-10 00:43:36.980000', -20.0]]
['IAQ/IRR Temp_adc_temperature', ['2016-11-10 00:43:36.980000', '2737']]
['IAQ/IRR Temp_adc_temperature', ['2016-11-10 00:43:36.980000', '2737']]
['Chemsense_no2', ['2016-11-10 00:43:36.980000', 554.0]]
['BMP180_temperature', ['2016-11-10 00:43:36.980000', 27.6]]
['BMP180_temperature', ['2016-11-10 00:43:36.980000', 27.6]]
['TSL260RD_intensity', ['2016-11-10 00:43:36.980000', 35.0]]
['TSL260RD_intensity', ['2016-11-10 00:43:36.980000', 35.0]]
['SHT25_humidity', ['2016-11-10 00:43:36.980000', 3001.0]]
['SHT25_humidity', ['2016-11-10 00:43:36.980000', 3001.0]]
['HTU21D_humidity', ['2016-11-10 00:43:36.980000', '18.47']]
['HTU21D_humidity', ['2016-11-10 00:43:36.980000', '18.47']]
['BMI160_orientation.y', ['2016-11-10 00:43:36.980000', 5.0]]
['BMI160_orientation.y', ['2016-11-10 00:43:36.980000', 5.0]]
['HMC5883L_magnetic_field.z', ['2016-11-10 00:43:36.980000', '0.22']]
['HMC5883L_magnetic_field.z', ['2016-11-10 00:43:36.980000', '0.22']]
['TSL250RD_intensity', ['2016-11-10 00:43:36.980000', 51.0]]
['TSL250RD_intensity', ['2016-11-10 00:43:36.980000', 51.0]]
['Coresense ID_mac_address', ['2016-11-10 00:43:36.980000', '00001814D5B0']]
['Coresense ID_mac_address', ['2016-11-10 00:43:36.980000', '00001814D5B0']]
['O3/NO2 Temp_adc_temperature', ['2016-11-10 00:43:36.980000', 2808.0]]
['O3/NO2 Temp_adc_temperature', ['2016-11-10 00:43:36.980000', 2808.0]]
['BMP180_pressure', ['2016-11-10 00:43:36.980000', 99954.0]]
['BMP180_pressure', ['2016-11-10 00:43:36.980000', 99954.0]]
['HTU21D_temperature', ['2016-11-10 00:43:36.980000', '26.08']]
['HTU21D_temperature', ['2016-11-10 00:43:36.980000', '26.08']]
['MMA8452Q_rms', ['2016-11-10 00:43:36.980000', '14.28']]
['MMA8452Q_rms', ['2016-11-10 00:43:36.980000', '14.28']]
['BMI160_acceleration.x', ['2016-11-10 00:43:36.980000', -4.0]]
['BMI160_acceleration.x', ['2016-11-10 00:43:36.980000', -4.0]]
['SO2/H2S Temp_adc_temperature', ['2016-11-10 00:43:36.980000', 2868.0]]
```

5. Now we need to start creating the stack on this profile and monitor for events to be created.

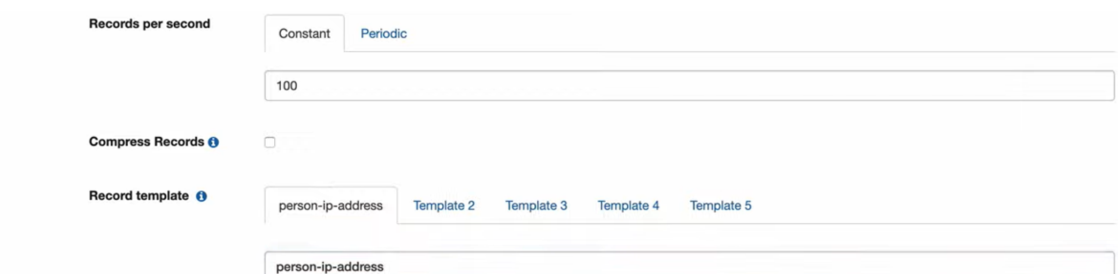


Kinesis-Data-Generator-Cognito-User
2022-03-28 16:04:13 UTC-0400
CREATE_IN_PROGRESS

Events (5)

Timestamp	Logical ID	Status	Status reason
2022-03-28 16:04:19 UTC-0400	AuthenticatedUserRole	CREATE_IN_PROGRESS	Resource creation Initiated
2022-03-28 16:04:18 UTC-0400	UnauthenticatedUserRole	CREATE_IN_PROGRESS	Resource creation Initiated
2022-03-28 16:04:18 UTC-0400	AuthenticatedUserRole	CREATE_IN_PROGRESS	-
2022-03-28 16:04:18 UTC-0400	UnauthenticatedUserRole	CREATE_IN_PROGRESS	-
2022-03-28 16:04:13 UTC-0400	Kinesis-Data-Generator-Cognito-User	CREATE_IN_PROGRESS	User Initiated

6. Now we need to start passing the .json file onto the stack that we had created on our KDG.
No we need to be quick because as soon as the data is released the entire burst will come onto the stack and we will need to stop it quick.



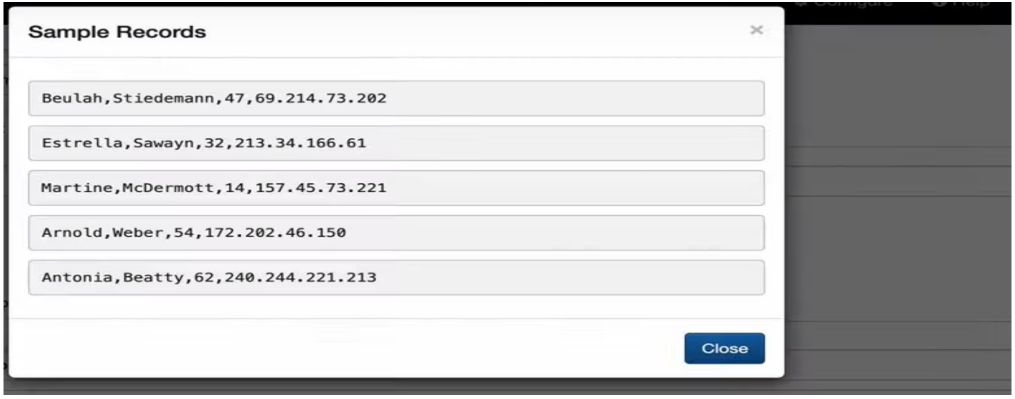
Records per second: Constant 100

Compress Records □

Record template: person-ip-address

person-ip-address

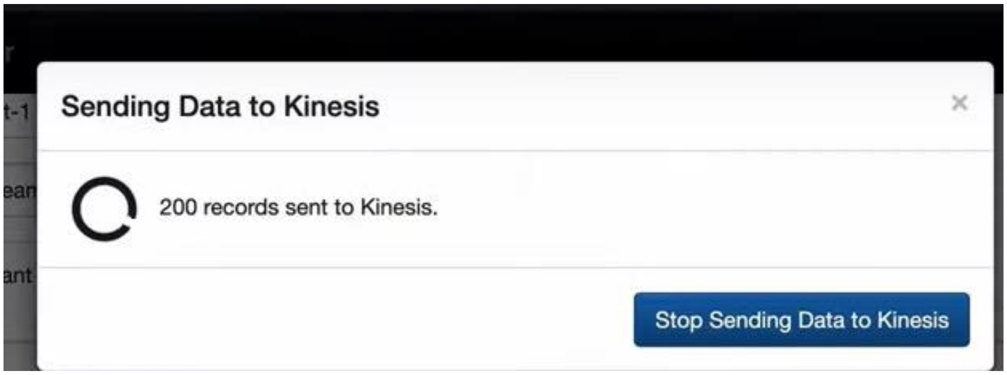
Passing a Valid input depending upon the test file in exercise.



Sample Records

```
Beulah,Stiedemann,47,69.214.73.202
Estrella,Sawayn,32,213.34.166.61
Martine,McDermott,14,157.45.73.221
Arnold,Weber,54,172.202.46.150
Antonia,Beatty,62,240.244.221.213
```

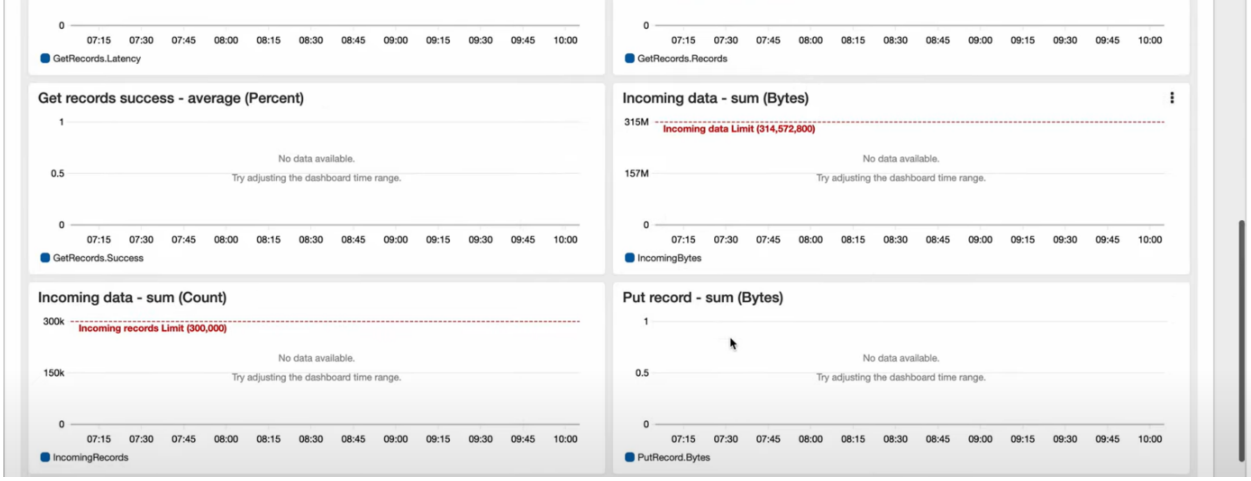
Tracking the progress to monitor the charts later.



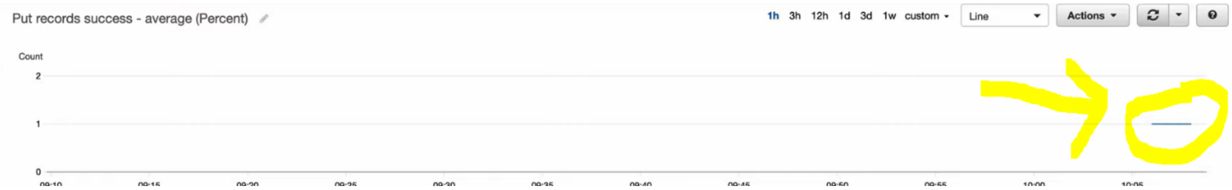
Sending Data to Kinesis

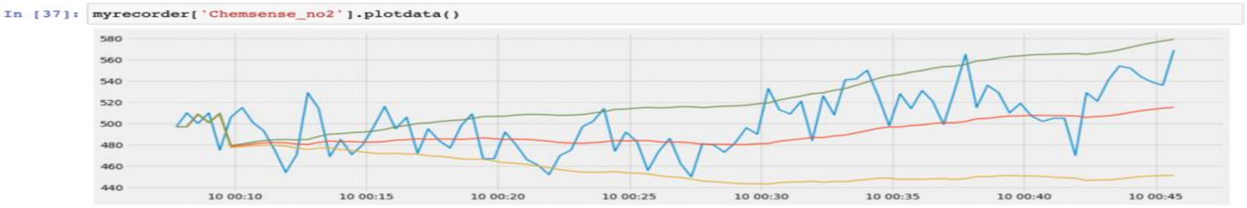
200 records sent to Kinesis.

7. Now we need to be quick as time is money. The burst has begin and we will soon see the influx of data in the empty charts below.



Just to check the influx of data we will need to one of the chart to see if the data overlays properly onto this stack.

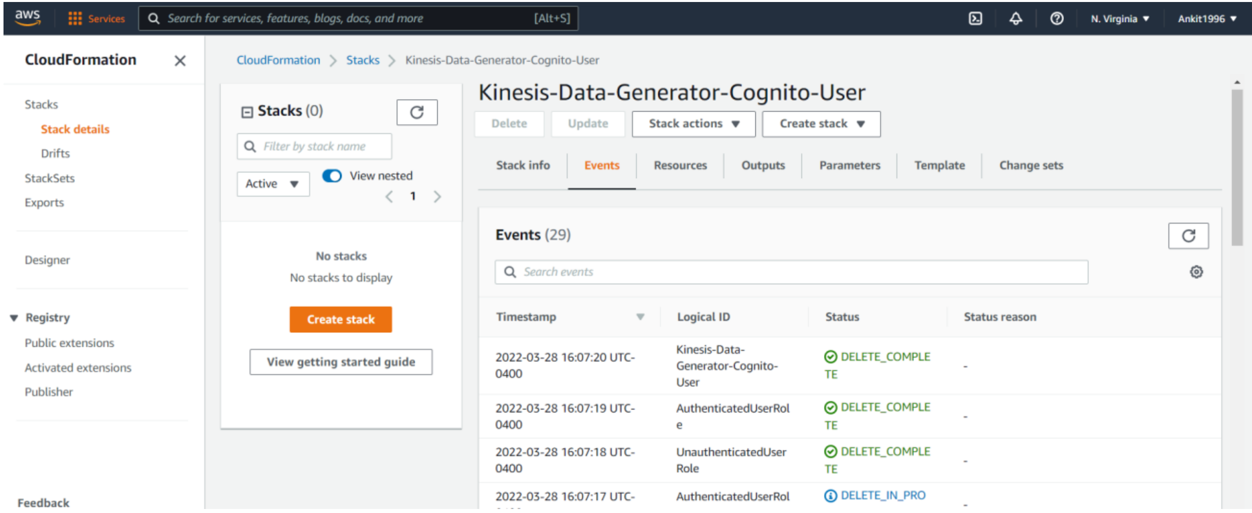




```
In [37]: myrecorder['Chemsense_no2'].plotdata()
```

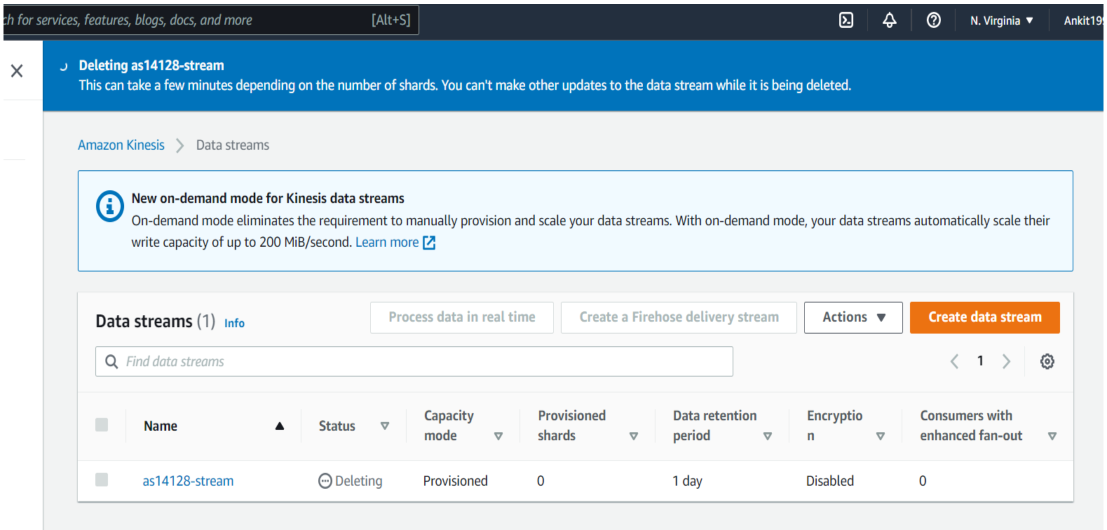
Since this method cost money for as long as it stays up, I have taken it down within a minute post recording the **completion** of the activity.

8. Finally I will be deleting the resources from KDG first.



Timestamp	Logical ID	Status	Status reason
2022-03-28 16:07:20 UTC-0400	Kinesis-Data-Generator-Cognito-User	DELETE_COMPLETE	-
2022-03-28 16:07:19 UTC-0400	AuthenticatedUserRole	DELETE_COMPLETE	-
2022-03-28 16:07:18 UTC-0400	UnauthenticatedUserRole	DELETE_COMPLETE	-
2022-03-28 16:07:17 UTC-	AuthenticatedUserRol	DELETE_IN_PRO	-

9. Then finally, I will delete the main Kinesis Datastream as well to cut down any further resource utilization.


h for services, features, blogs, docs, and more [Alt+S]
N. Virginia
Ankit19

Deleting as14128-stream
This can take a few minutes depending on the number of shards. You can't make other updates to the data stream while it is being deleted.

Amazon Kinesis > Data streams

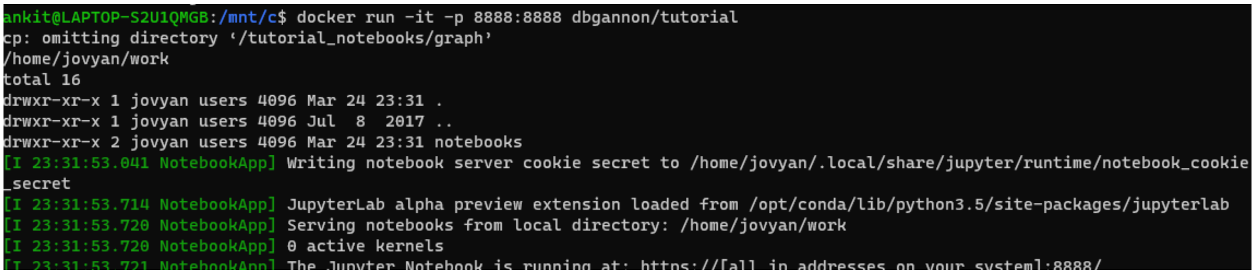
New on-demand mode for Kinesis data streams
On-demand mode eliminates the requirement to manually provision and scale your data streams. With on-demand mode, your data streams automatically scale their write capacity of up to 200 MiB/second. Learn more

Name	Status	Capacity mode	Provisioned shards	Data retention period	Encryption	Consumers with enhanced fan-out
as14128-stream	Deleting	Provisioned	0	1 day	Disabled	0

Part III: Google Datalab

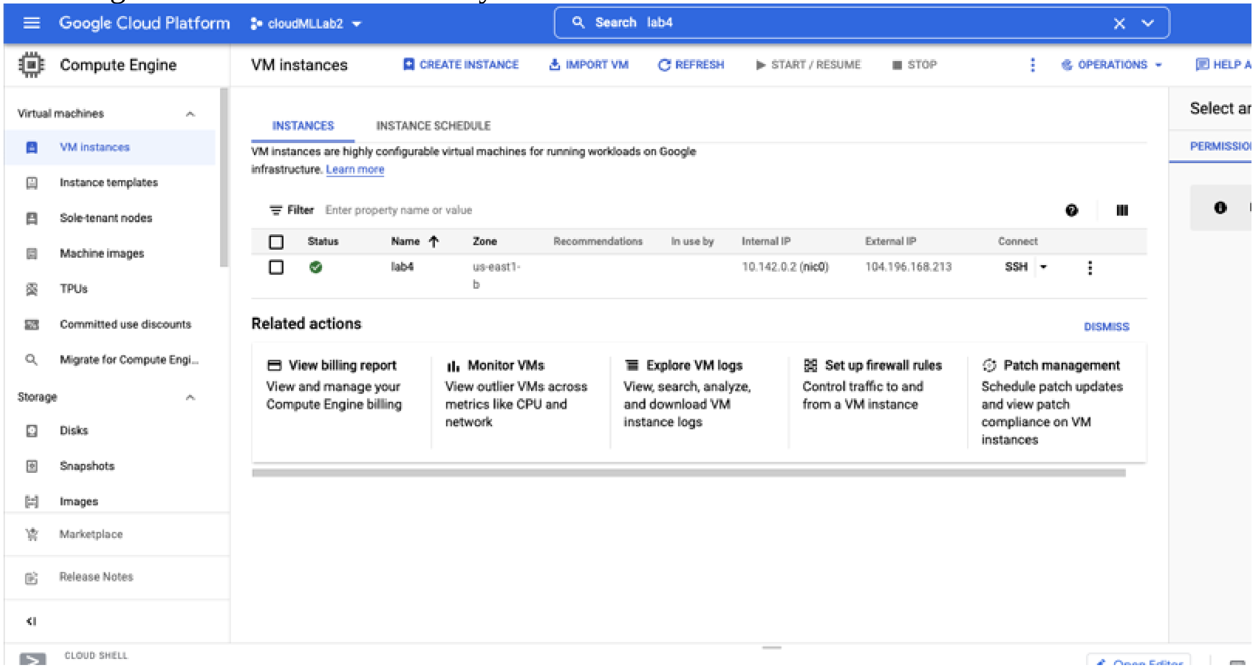
1. We need to setup both of the notebooks on the google labs CLI.

   a. Notebook 1



```
ankit@LAPTOP-S2U1QMGB:/mnt/c$ docker run -it -p 8888:8888 dbgannon/tutorial
cp: omitting directory ‘/tutorial_notebooks/graph’
/home/jovyan/work
total 16
drwxr-xr-x 1 jovyan users 4096 Mar 24 23:31 .
drwxr-xr-x 1 jovyan users 4096 Jul  8  2017 ..
drwxr-xr-x 2 jovyan users 4096 Mar 24 23:31 notebooks
[I 23:31:53.041 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie
_secret
[I 23:31:53.714 NotebookApp] JupyterLab alpha preview extension loaded from /opt/conda/lib/python3.5/site-packages/jupyterlab
[I 23:31:53.720 NotebookApp] Serving notebooks from local directory: /home/jovyan/work
[I 23:31:53.720 NotebookApp] 0 active kernels
[I 23:31:53.721 NotebookApp] The Jupyter Notebook is running at: https://[all ip addresses on your system]:8888/
```

   b. Checking both the notebooks in the system GUI.



VM instances are highly configurable virtual machines for running workloads on Google infrastructure. Learn more

☐	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
☐	✓	lab4	us-east1-b			10.142.0.2 (nic0)	104.196.168.213	SSH

Related actions

DISMISS

View billing report
View and manage your Compute Engine billing

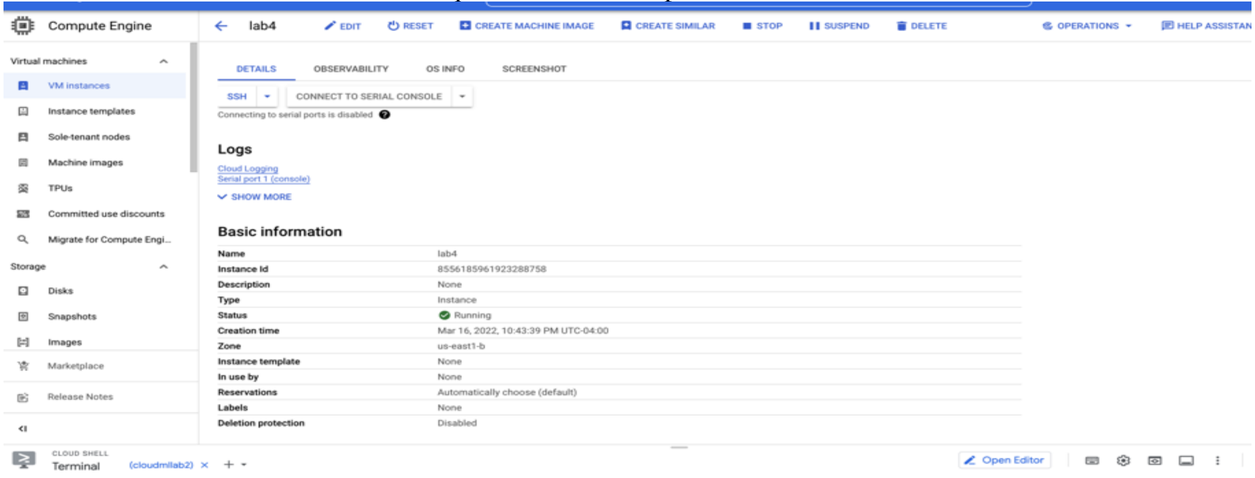
Monitor VMs
View outlier VMs across metrics like CPU and network

Explore VM logs
View, search, analyze, and download VM instance logs

Set up firewall rules
Control traffic to and from a VM instance

Patch management
Schedule patch updates and view patch compliance on VM instances

   c. Check the basic lab information required for this experiment.



Connecting to serial ports is disabled

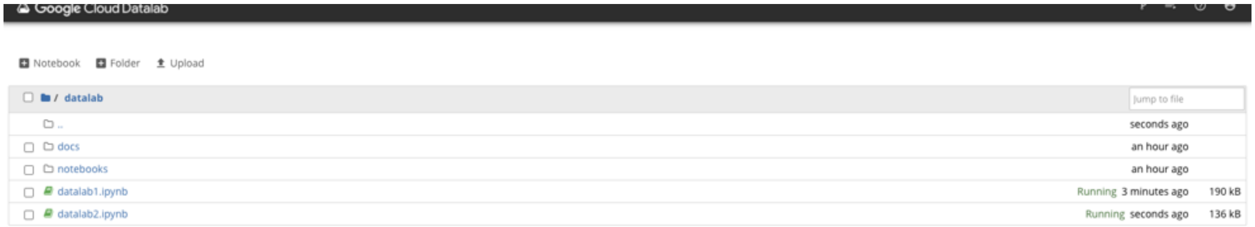
Logs

Cloud Logging
Serial port 1 (console)
SHOW MORE

Basic information

Name	lab4
Instance Id	8556185961923288758
Description	None
Type	Instance
Status	✓ Running
Creation time	Mar 16, 2022, 10:43:39 PM UTC-04:00
Zone	us-east1-b
Instance template	None
In use by	None
Reservations	Automatically choose (default)
Labels	None
Deletion protection	Disabled

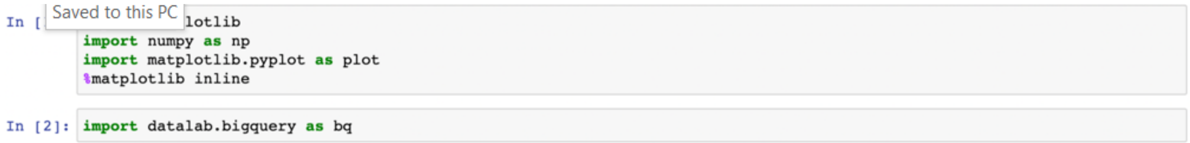
d. Finally uploading both the datalab notebooks to the google cloud.



□ / datalab			
..		seconds ago	
□ docs		an hour ago	
□ notebooks		an hour ago	
□ datalab1.ipynb	Running	3 minutes ago	190 kB
□ datalab2.ipynb	Running	seconds ago	136 kB

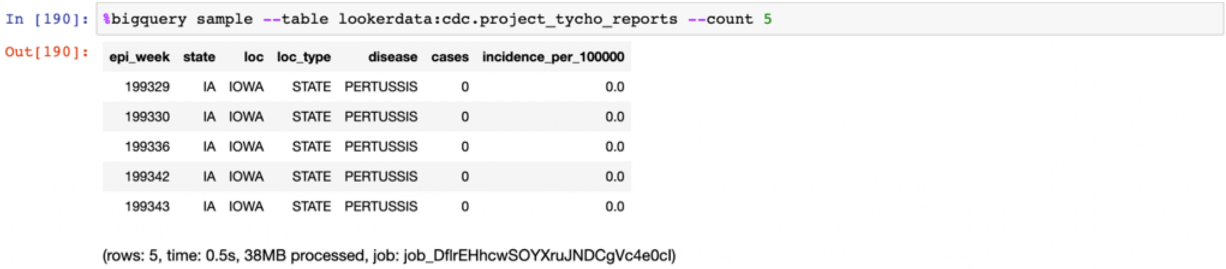
- **Datalab I**

  In this first lab we are trying to track the rate of spread between the two cities that are separated over a distance of 2000 miles.
  Having said that we need to run this on a single notepook.



```
In [        lotlib
        import numpy as np
        import matplotlib.pyplot as plot
        %matplotlib inline

In [2]: import datalab.bigquery as bq
```

Checking the same inputs in the reports before commit.

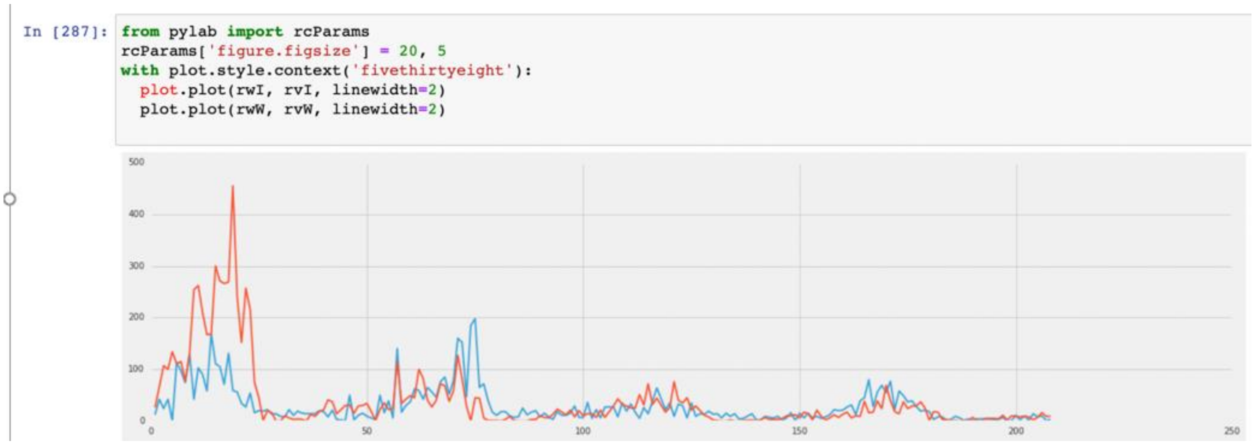


```
In [190]: %bigquery sample --table lookerdata:cdc.project_tycho_reports --count 5
Out[190]:
```

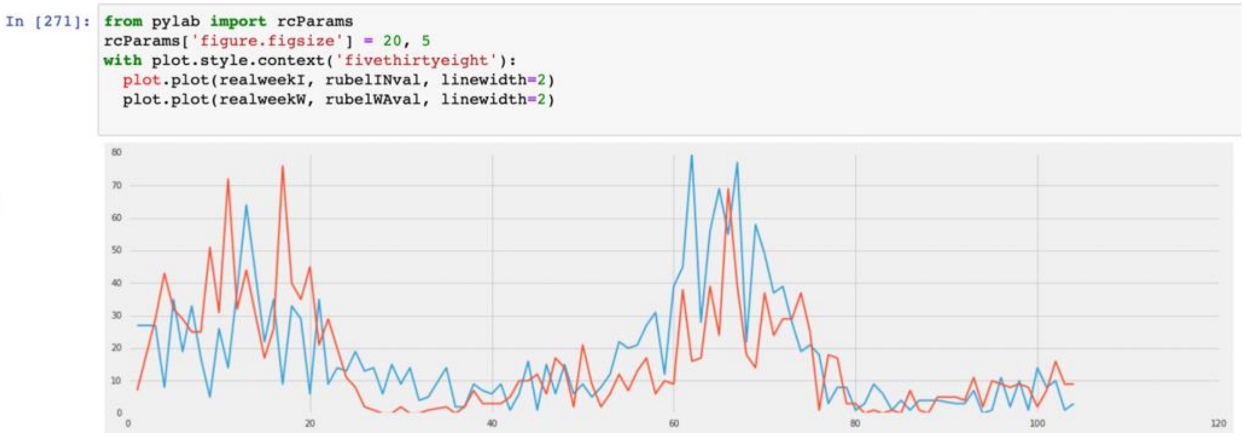
epi_week	state	loc	loc_type	disease	cases	incidence_per_100000
199329	IA	IOWA	STATE	PERTUSSIS	0	0.0
199330	IA	IOWA	STATE	PERTUSSIS	0	0.0
199336	IA	IOWA	STATE	PERTUSSIS	0	0.0
199342	IA	IOWA	STATE	PERTUSSIS	0	0.0
199343	IA	IOWA	STATE	PERTUSSIS	0	0.0

(rows: 5, time: 0.5s, 38MB processed, job: job_DflrEHhcwSOYXruJNDCgVc4e0cl)

Now we need to track the activity on the graph by importing the "**rcParams**" from the same pyLab.



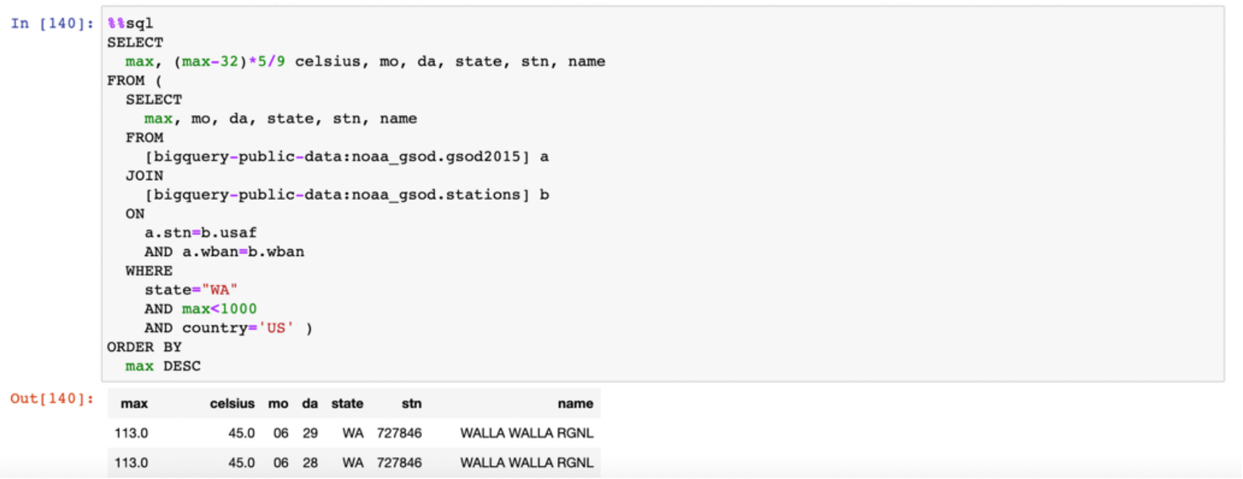
```
In [287]: from pylab import rcParams
          rcParams['figure.figsize'] = 20, 5
          with plot.style.context('fivethirtyeight'):
            plot.plot(rwI, rvI, linewidth=2)
            plot.plot(rwW, rvW, linewidth=2)
```



```
In [271]: from pylab import rcParams
          rcParams['figure.figsize'] = 20, 5
          with plot.style.context('fivethirtyeight'):
            plot.plot(realweekI, rubelINval, linewidth=2)
            plot.plot(realweekW, rubelWAval, linewidth=2)
```

Next we need to compare the two years. The data stored in lab 1 , will go through the same normalization and plot the boundaries for the next 4 years.

- **Datalab II** – Abnormality in the weather stations in Washington.

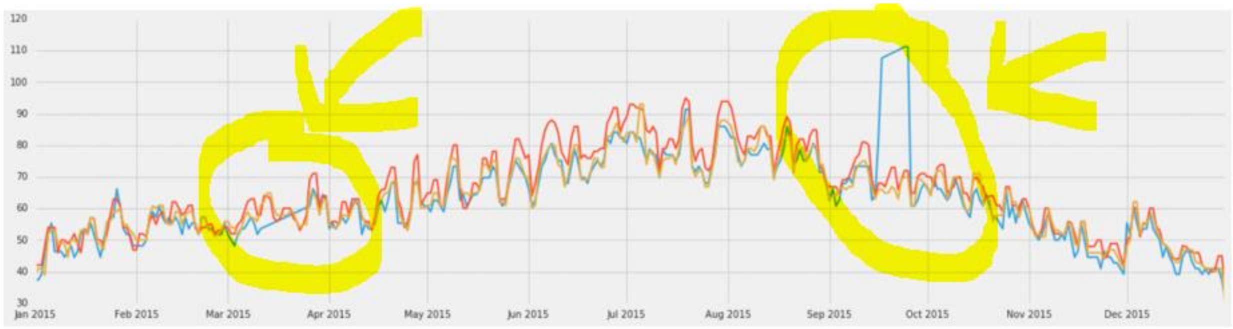


```
In [140]: %%sql
          SELECT
            max, (max-32)*5/9 celsius, mo, da, state, stn, name
          FROM (
            SELECT
              max, mo, da, state, stn, name
            FROM
              [bigquery-public-data:noaa_gsod.gsod2015] a
            JOIN
              [bigquery-public-data:noaa_gsod.stations] b
            ON
              a.stn=b.usaf
              AND a.wban=b.wban
            WHERE
              state="WA"
              AND max<1000
              AND country='US' )
          ORDER BY
            max DESC
```

Out[140]:

max	celsius	mo	da	state	stn	name
113.0	45.0	06	29	WA	727846	WALLA WALLA RGNL
113.0	45.0	06	28	WA	727846	WALLA WALLA RGNL

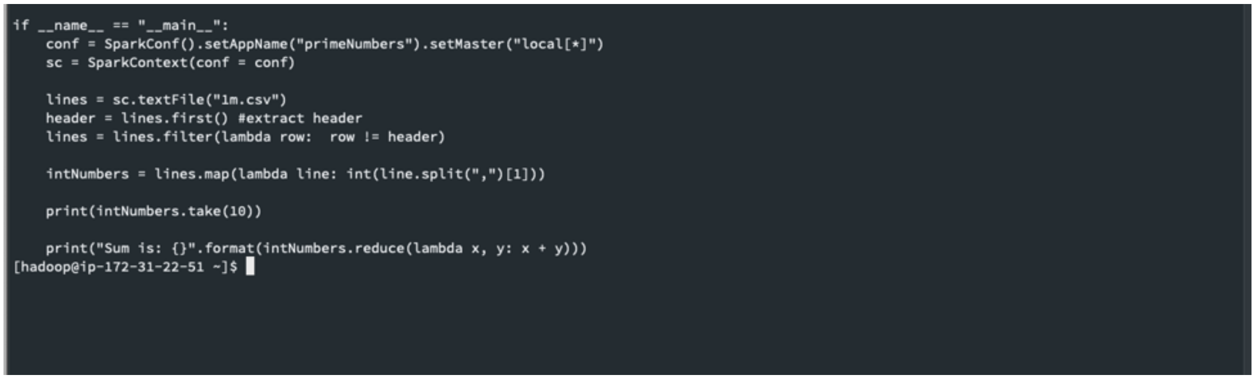
By tracing the below graph, we can see that the skagit station is the culprit.
Apart from this we can see the anomalies in the month of September and March for the same year.



Part VI: Google Datalab

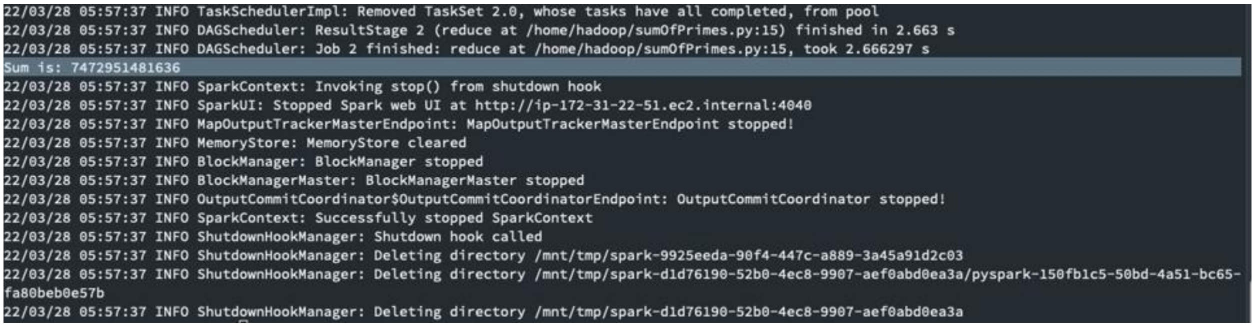
This experiment was done on an EMR Hadoop cluster as well. A program was executed as a spark batch processing job to count the sum of the first 1 million prime numbers

Python code used is mentioned below.



```
if __name__ == "__main__":
    conf = SparkConf().setAppName("primeNumbers").setMaster("local[*]")
    sc = SparkContext(conf = conf)

    lines = sc.textFile("1m.csv")
    header = lines.first() #extract header
    lines = lines.filter(lambda row:  row != header)

    intNumbers = lines.map(lambda line: int(line.split(",")[1]))

    print(intNumbers.take(10))

    print("Sum is: {}".format(intNumbers.reduce(lambda x, y: x + y)))
[hadoop@ip-172-31-22-51 ~]$ 
```

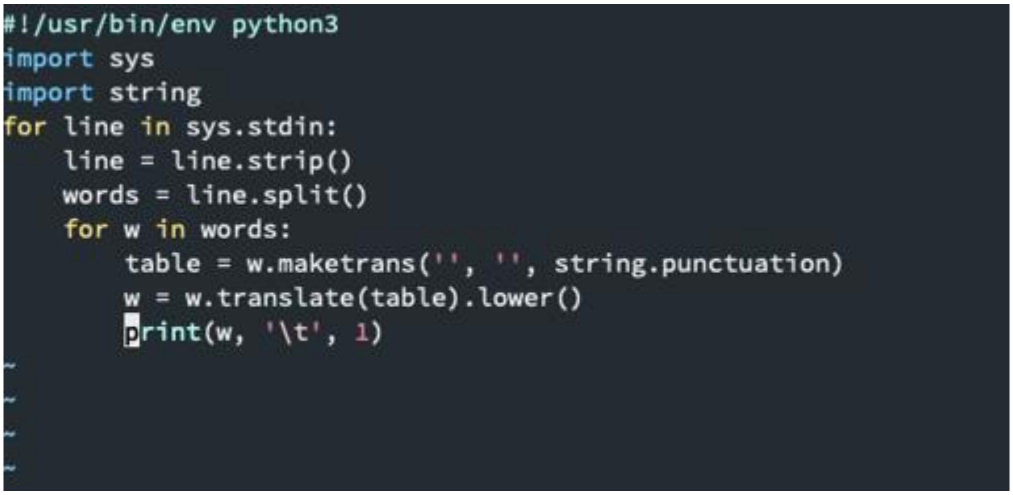
Final build and answer of the above .py file.



```
22/03/28 05:57:37 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
22/03/28 05:57:37 INFO DAGScheduler: ResultStage 2 (reduce at /home/hadoop/sumOfPrimes.py:15) finished in 2.663 s
22/03/28 05:57:37 INFO DAGScheduler: Job 2 finished: reduce at /home/hadoop/sumOfPrimes.py:15, took 2.666297 s
Sum is: 7472951481636
22/03/28 05:57:37 INFO SparkContext: Invoking stop() from shutdown hook
22/03/28 05:57:37 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-22-51.ec2.internal:4040
22/03/28 05:57:37 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/03/28 05:57:37 INFO MemoryStore: MemoryStore cleared
22/03/28 05:57:37 INFO BlockManager: BlockManager stopped
22/03/28 05:57:37 INFO BlockManagerMaster: BlockManagerMaster stopped
22/03/28 05:57:37 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/03/28 05:57:37 INFO SparkContext: Successfully stopped SparkContext
22/03/28 05:57:37 INFO ShutdownHookManager: Shutdown hook called
22/03/28 05:57:37 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-9925eeda-90f4-447c-a889-3a45a91d2c03
22/03/28 05:57:37 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-d1d76190-52b0-4ec8-9907-aef0abd0ea3a/pyspark-150fb1c5-50bd-4a51-bc65-
fa80beb0e57b
22/03/28 05:57:37 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-d1d76190-52b0-4ec8-9907-aef0abd0ea3a
```

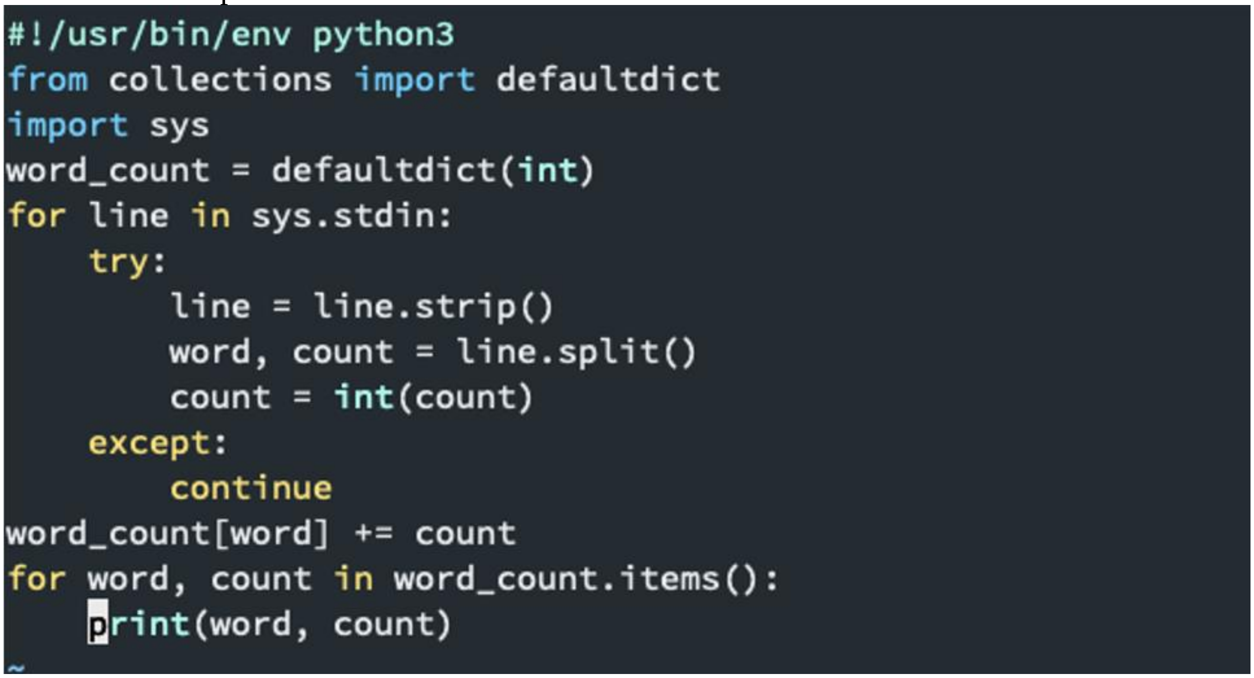
Part VII: Build additional experiments to document the use of MapReduce using Hadoop.

1. Create a generic mapper function that calculates the number of words in the text file put in hdfs.



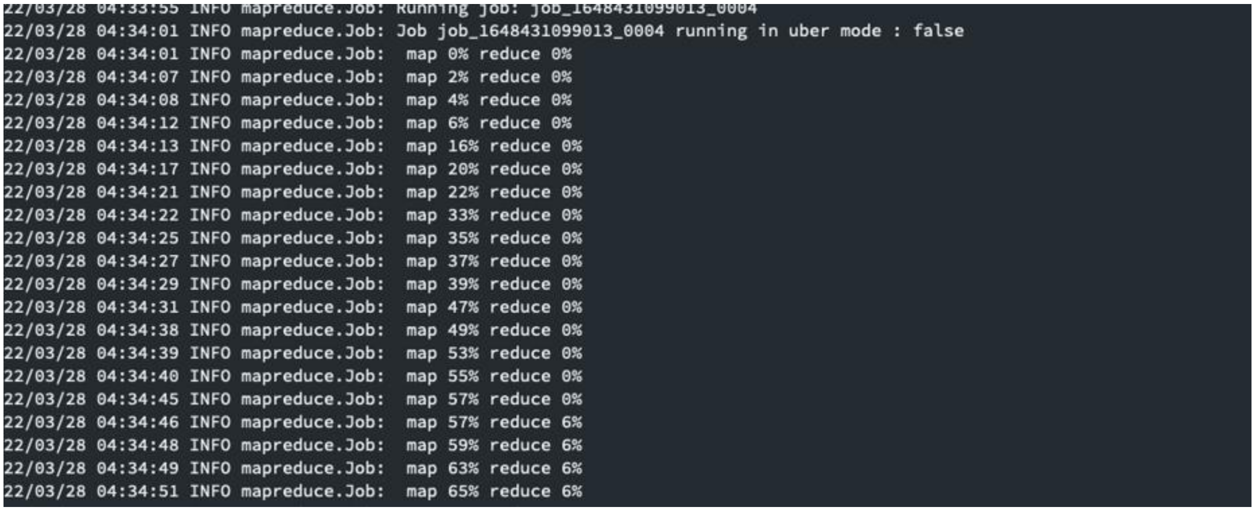
```
#!/usr/bin/env python3
import sys
import string
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for w in words:
        table = w.maketrans('', '', string.punctuation)
        w = w.translate(table).lower()
        print(w, '\t', 1)
```

2. Now we need to put the reducer file to count the number of words in the document.



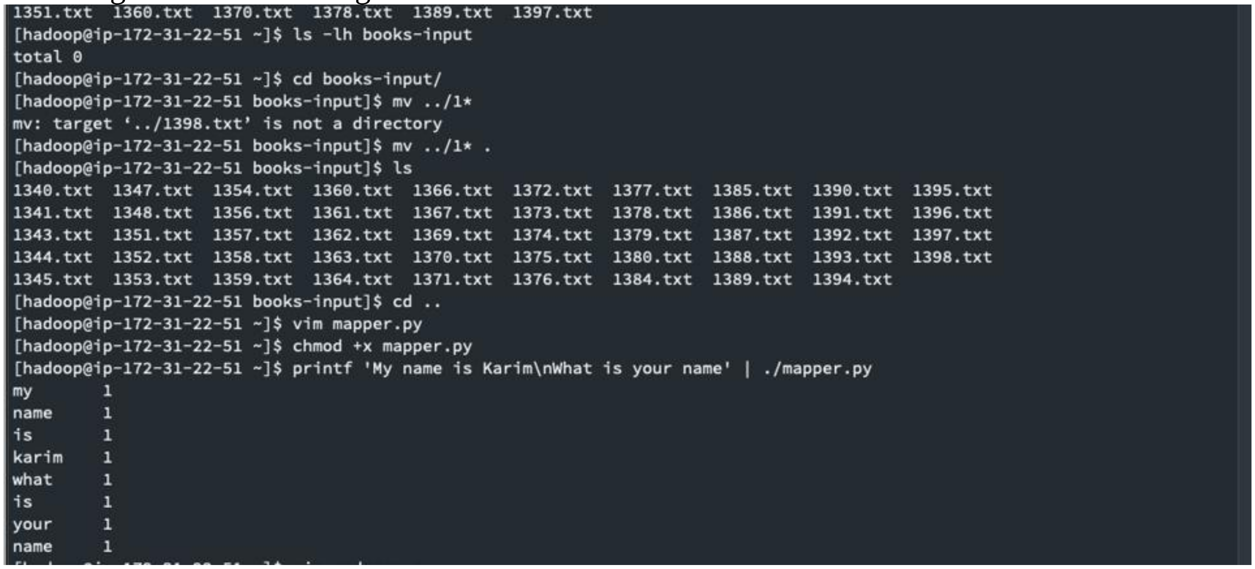
```
#!/usr/bin/env python3
from collections import defaultdict
import sys
word_count = defaultdict(int)
for line in sys.stdin:
    try:
        line = line.strip()
        word, count = line.split()
        count = int(count)
    except:
        continue
word_count[word] += count
for word, count in word_count.items():
    print(word, count)
```

3. Submit both the application and monitor the status of map reduce over the files.



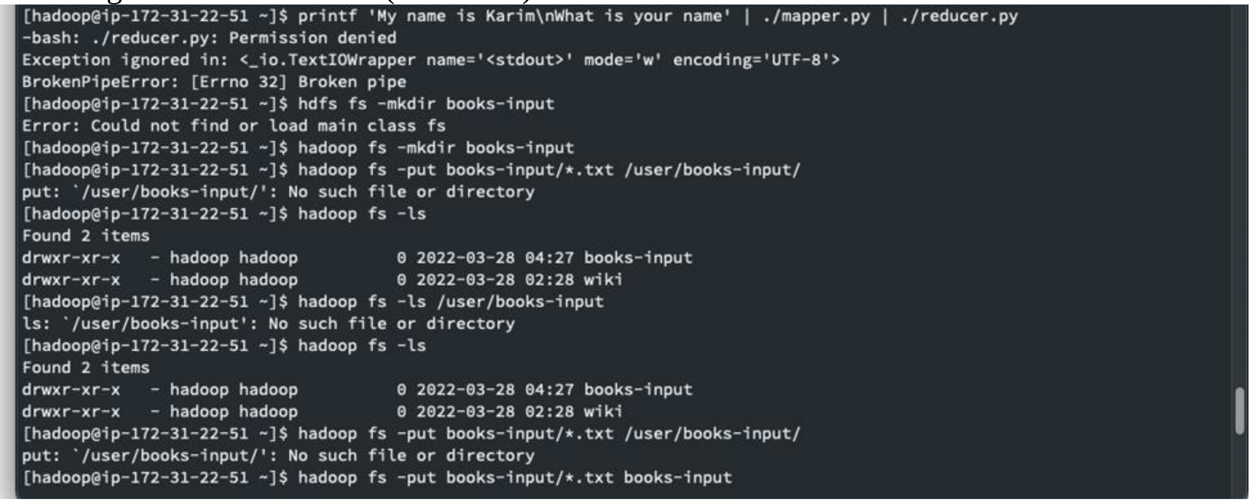
```
22/03/28 04:33:55 INFO mapreduce.Job: Running job: job_1648431099013_0004
22/03/28 04:34:01 INFO mapreduce.Job: Job job_1648431099013_0004 running in uber mode : false
22/03/28 04:34:01 INFO mapreduce.Job:  map 0% reduce 0%
22/03/28 04:34:07 INFO mapreduce.Job:  map 2% reduce 0%
22/03/28 04:34:08 INFO mapreduce.Job:  map 4% reduce 0%
22/03/28 04:34:12 INFO mapreduce.Job:  map 6% reduce 0%
22/03/28 04:34:13 INFO mapreduce.Job:  map 16% reduce 0%
22/03/28 04:34:17 INFO mapreduce.Job:  map 20% reduce 0%
22/03/28 04:34:21 INFO mapreduce.Job:  map 22% reduce 0%
22/03/28 04:34:22 INFO mapreduce.Job:  map 33% reduce 0%
22/03/28 04:34:25 INFO mapreduce.Job:  map 35% reduce 0%
22/03/28 04:34:27 INFO mapreduce.Job:  map 37% reduce 0%
22/03/28 04:34:29 INFO mapreduce.Job:  map 39% reduce 0%
22/03/28 04:34:31 INFO mapreduce.Job:  map 47% reduce 0%
22/03/28 04:34:38 INFO mapreduce.Job:  map 49% reduce 0%
22/03/28 04:34:39 INFO mapreduce.Job:  map 53% reduce 0%
22/03/28 04:34:40 INFO mapreduce.Job:  map 55% reduce 0%
22/03/28 04:34:45 INFO mapreduce.Job:  map 57% reduce 0%
22/03/28 04:34:46 INFO mapreduce.Job:  map 57% reduce 6%
22/03/28 04:34:48 INFO mapreduce.Job:  map 59% reduce 6%
22/03/28 04:34:49 INFO mapreduce.Job:  map 63% reduce 6%
22/03/28 04:34:51 INFO mapreduce.Job:  map 65% reduce 6%
```

4. Checking the count of a single sentence.



```
1351.txt  1360.txt  1370.txt  1378.txt  1389.txt  1397.txt
[hadoop@ip-172-31-22-51 ~]$ ls -lh books-input
total 0
[hadoop@ip-172-31-22-51 ~]$ cd books-input/
[hadoop@ip-172-31-22-51 books-input]$ mv ../1*
mv: target ‘../1398.txt’ is not a directory
[hadoop@ip-172-31-22-51 books-input]$ mv ../1* .
[hadoop@ip-172-31-22-51 books-input]$ ls
1340.txt  1347.txt  1354.txt  1360.txt  1366.txt  1372.txt  1377.txt  1385.txt  1390.txt  1395.txt
1341.txt  1348.txt  1356.txt  1361.txt  1367.txt  1373.txt  1378.txt  1386.txt  1391.txt  1396.txt
1343.txt  1351.txt  1357.txt  1362.txt  1369.txt  1374.txt  1379.txt  1387.txt  1392.txt  1397.txt
1344.txt  1352.txt  1358.txt  1363.txt  1370.txt  1375.txt  1380.txt  1388.txt  1393.txt  1398.txt
1345.txt  1353.txt  1359.txt  1364.txt  1371.txt  1376.txt  1384.txt  1389.txt  1394.txt
[hadoop@ip-172-31-22-51 books-input]$ cd ..
[hadoop@ip-172-31-22-51 ~]$ vim mapper.py
[hadoop@ip-172-31-22-51 ~]$ chmod +x mapper.py
[hadoop@ip-172-31-22-51 ~]$ printf 'My name is Karim\nWhat is your name' | ./mapper.py
my      1
name    1
is      1
karim   1
what    1
is      1
your    1
name    1
```

5. Matching results of the book. (Initial file)



```
[hadoop@ip-172-31-22-51 ~]$ printf 'My name is Karim\nWhat is your name' | ./mapper.py | ./reducer.py
-bash: ./reducer.py: Permission denied
Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>
BrokenPipeError: [Errno 32] Broken pipe
[hadoop@ip-172-31-22-51 ~]$ hdfs fs -mkdir books-input
Error: Could not find or load main class fs
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -mkdir books-input
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -put books-input/*.txt /user/books-input/
put: `/user/books-input/': No such file or directory
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2022-03-28 04:27 books-input
drwxr-xr-x   - hadoop hadoop          0 2022-03-28 02:28 wiki
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -ls /user/books-input
ls: `/user/books-input': No such file or directory
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2022-03-28 04:27 books-input
drwxr-xr-x   - hadoop hadoop          0 2022-03-28 02:28 wiki
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -put books-input/*.txt /user/books-input/
put: `/user/books-input/': No such file or directory
[hadoop@ip-172-31-22-51 ~]$ hadoop fs -put books-input/*.txt books-input
```