Project 4 : Stack Overflow - askubuntu dataset

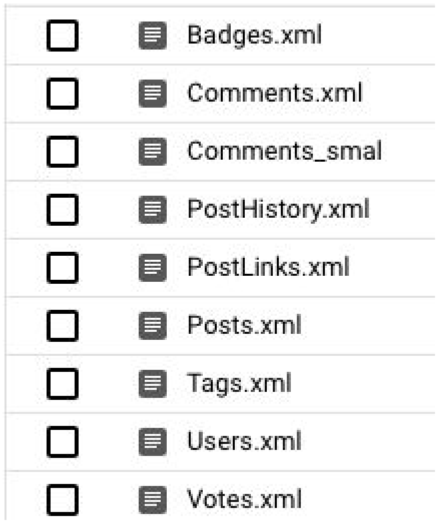
Adarsh and Ayush

The project involves extracting data from several smaller datasets and combining them together to do analysis.

Dataset:

The dataset contains logs for askubuntu stackexchange logs (https://askubuntu.com) in XML format.

- The total size of the dataset is 22 GB.
- It is stored in the GCS bucket gs://stackoverflow-dataset-677
- The dataset includes multiple xml files corresponding to different attributes of the dataset.



Badges.xml
Comments.xml
Comments_smal
PostHistory.xml
PostLinks.xml
Posts.xml
Tags.xml
Users.xml
Votes.xml

- The following are the relevant features for each XML file.
    - **Users**: Reputation, CreationDate, DisplayName, WebsiteUrl, Location, Views, UpVotes, DownVotes, AccountId
    - **Posts**: Id, PostTypeId, AcceptedAnswerId, CreationDate, Score, Body, OwnerUserId, Title, Tags, AnswerCount, CommentCount
    - **Comments** : RowId, PostId, Score, Text, CreationDate, UserId, ContentLicense
    - **Tags**: RowId, TagName, Count, ExcerptPostId, WikiPostId
    - **Badges** : RowId, UserId, Name, Date, Class, TagBased
    - **Votes** : RowId, PostId, VoteTypeId, CreationDate

Tasks to be completed:

1. a) Extracting user id and username and storing it to GCS.
   b) Extracting the comments and doing inner join with Spark SQL and display to user.

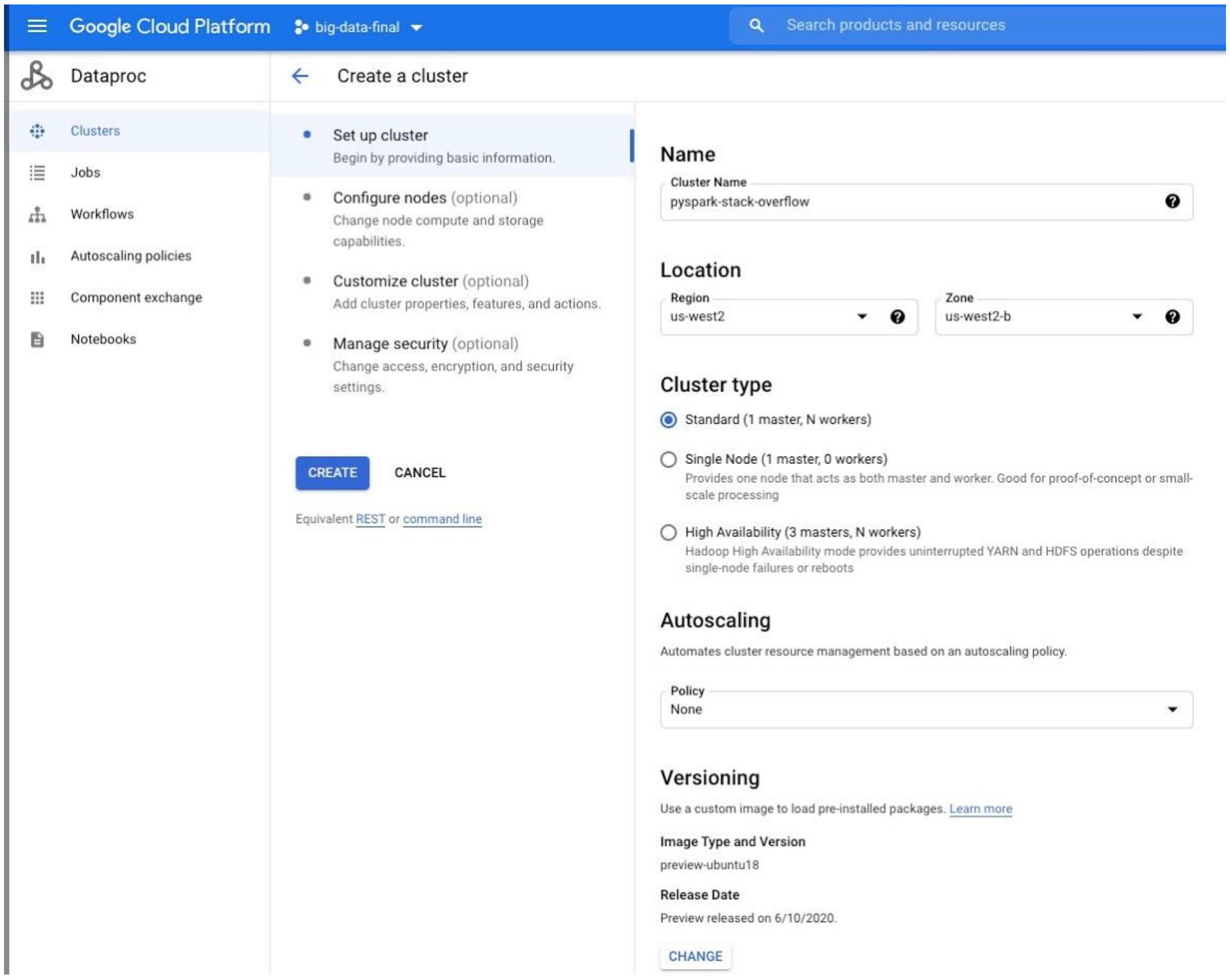
2. Trending Users with max comments: Finding users who posted maximum comments and visualizing it.
3. Trending Topics in Comments: Extracting the trending topics in comments related to ubuntu and visualizing it.
4. Automating task 1 with Apache Airflow i.e. in a single pipeline

Server Setup:

We used Google Cloud DataProc to setup a pyspark cluster of 1 Master Nodes (2 vCPUs) and 6 Worker Nodes (1vCPU) each with spark 3.2, python 3 and ubuntu 18.04 operating system.

For first 3 tasks, the server setup was done manually.

Manual Setup:



Create a cluster

Set up cluster
Begin by providing basic information.

Configure nodes (optional)
Change node compute and storage capabilities.

Customize cluster (optional)
Add cluster properties, features, and actions.

Manage security (optional)
Change access, encryption, and security settings.

Equivalent REST or command line

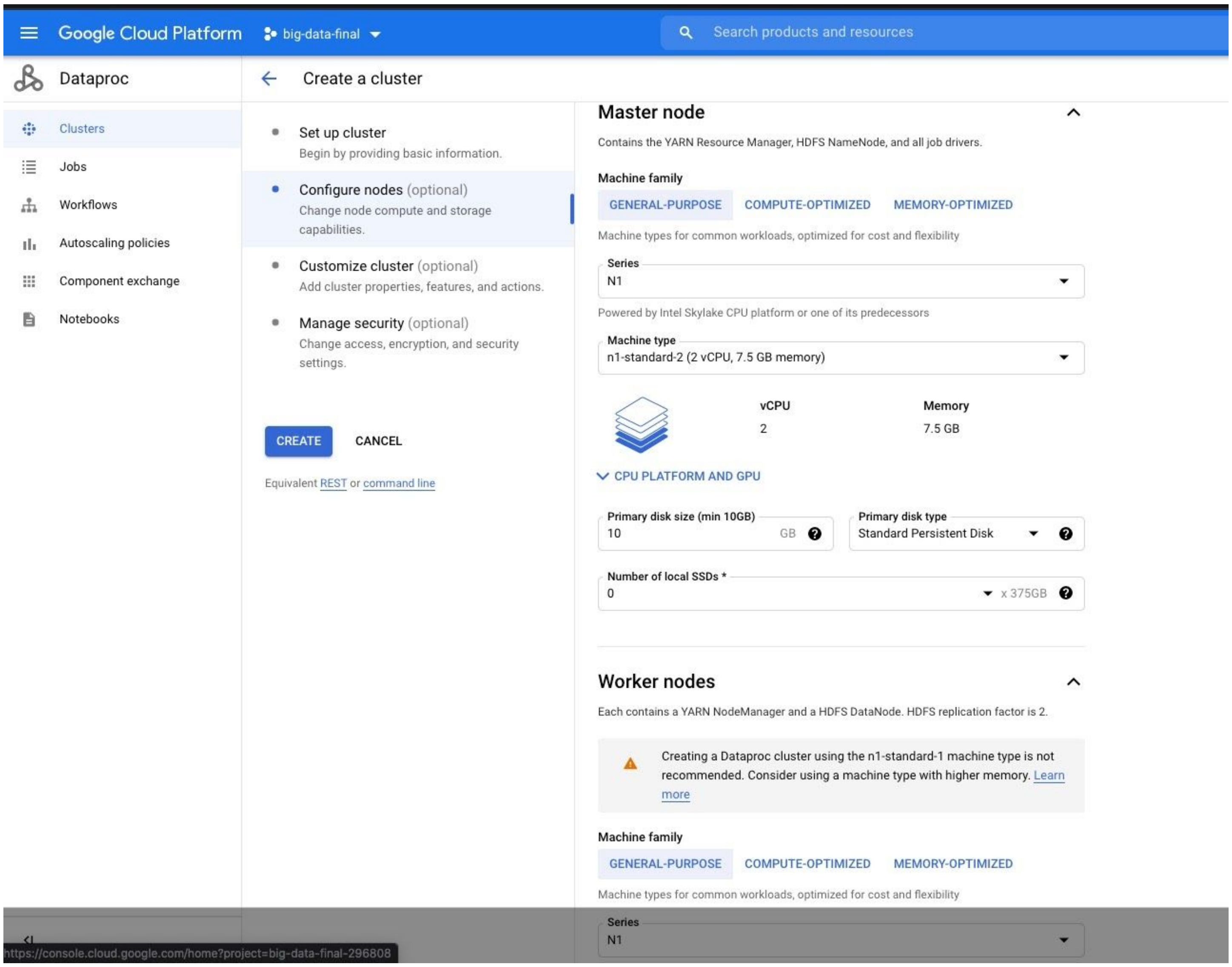
Name
Cluster Name: pyspark-stack-overflow

Location
Region: us-west2
Zone: us-west2-b

Cluster type
[✓ Standard (1 master, N workers)]
Single Node (1 master, 0 workers)
Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing
High Availability (3 masters, N workers)
Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Autoscaling
Automates cluster resource management based on an autoscaling policy.
Policy: None

Versioning
Use a custom image to load pre-installed packages. Learn more
Image Type and Version
preview-ubuntu18
Release Date
Preview released on 6/10/2020.


Google Cloud Platform
big-data-final
Search products and resources
Dataproc
Clusters
Jobs
Workflows
Autoscaling policies
Component exchange
Notebooks
Create a cluster
Set up cluster
Begin by providing basic information.
Configure nodes (optional)
Change node compute and storage capabilities.
Customize cluster (optional)
Add cluster properties, features, and actions.
Manage security (optional)
Change access, encryption, and security settings.
CREATE
CANCEL
Equivalent REST or command line

Master node

Contains the YARN Resource Manager, HDFS NameNode, and all job drivers.

Machine family: GENERAL-PURPOSE | COMPUTE-OPTIMIZED | MEMORY-OPTIMIZED

Machine types for common workloads, optimized for cost and flexibility

Series: N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type: n1-standard-2 (2 vCPU, 7.5 GB memory)

vCPU	Memory
2	7.5 GB

CPU PLATFORM AND GPU

Primary disk size (min 10GB): 10 GB

Primary disk type: Standard Persistent Disk

Number of local SSDs *: 0 x 375GB

Worker nodes

Each contains a YARN NodeManager and a HDFS DataNode. HDFS replication factor is 2.

Creating a Dataproc cluster using the n1-standard-1 machine type is not recommended. Consider using a machine type with higher memory. Learn more

Machine family: GENERAL-PURPOSE | COMPUTE-OPTIMIZED | MEMORY-OPTIMIZED

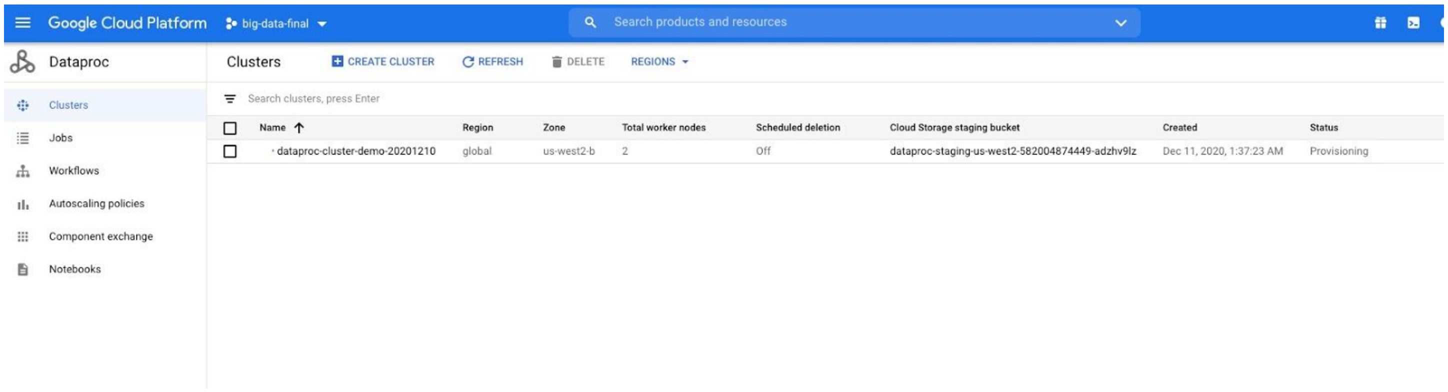
Machine types for common workloads, optimized for cost and flexibility

Series: N1

https://console.cloud.google.com/home?project=big-data-final-296808

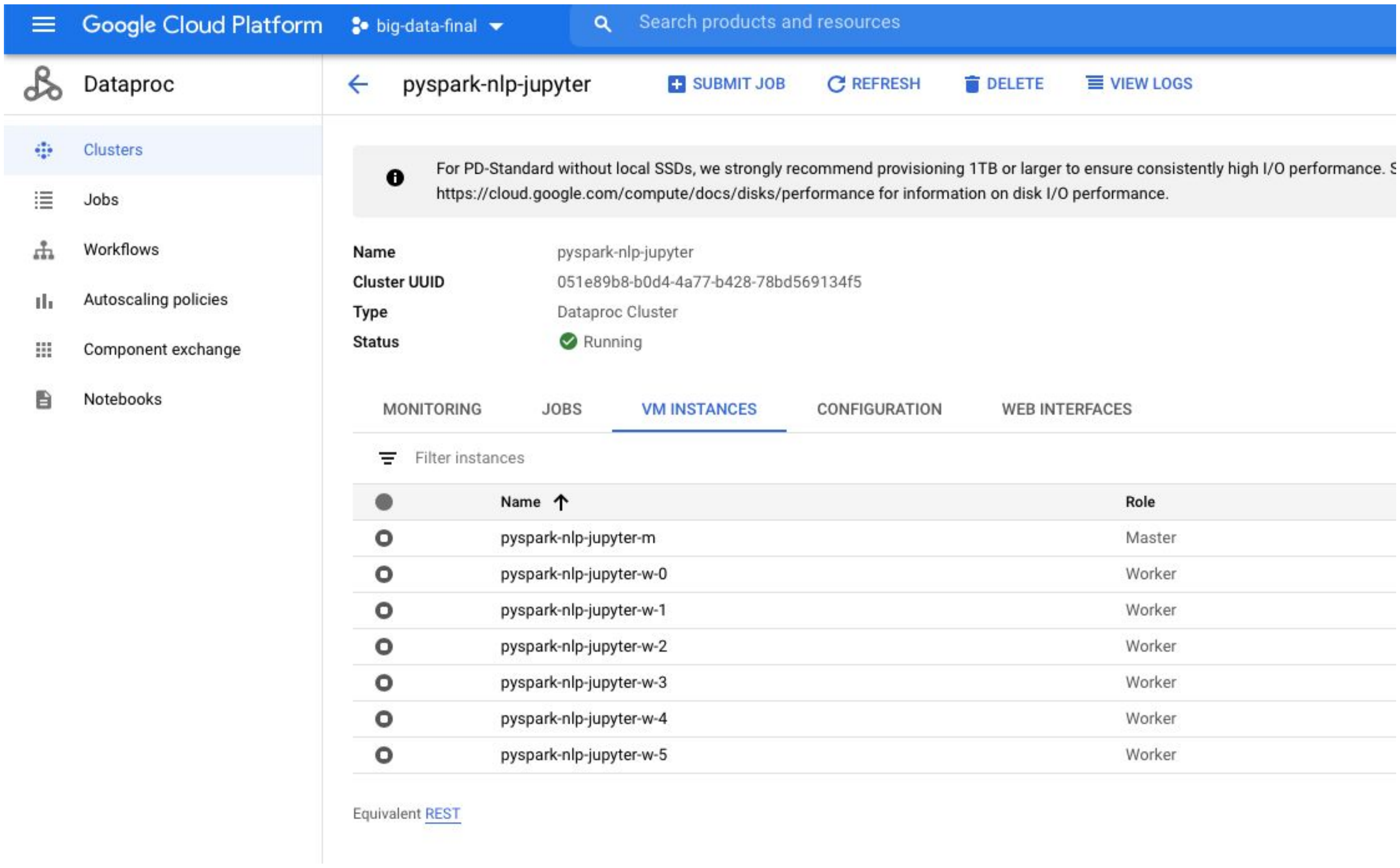
Result:

- **Cluster**



Name	Region	Zone	Total worker nodes	Scheduled deletion	Cloud Storage staging bucket	Created	Status
dataproc-cluster-demo-20201210	global	us-west2-b	2	Off	dataproc-staging-us-west2-582004874449-adzhv9lz	Dec 11, 2020, 1:37:23 AM	Provisioning

- **Cluster Nodes with 1 master and 6 worker nodes**

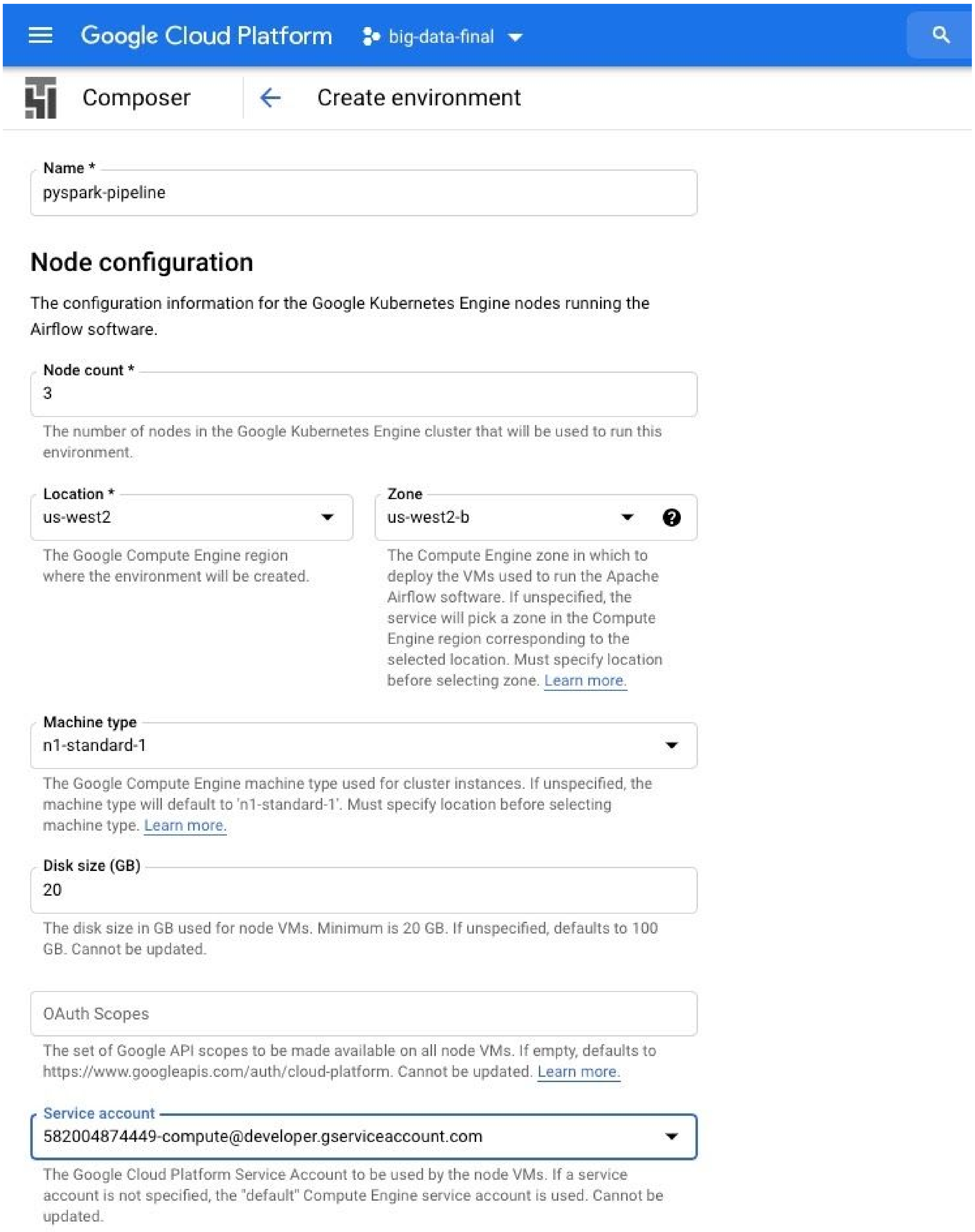


For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. S
https://cloud.google.com/compute/docs/disks/performance for information on disk I/O performance.

Name	pyspark-nlp-jupyter
Cluster UUID	051e89b8-b0d4-4a77-b428-78bd569134f5
Type	Dataproc Cluster
Status	Running

Name	Role
pyspark-nlp-jupyter-m	Master
pyspark-nlp-jupyter-w-0	Worker
pyspark-nlp-jupyter-w-1	Worker
pyspark-nlp-jupyter-w-2	Worker
pyspark-nlp-jupyter-w-3	Worker
pyspark-nlp-jupyter-w-4	Worker
pyspark-nlp-jupyter-w-5	Worker

For the last task i.e. automating task 1 with airflow, an airflow cluster was set up using Google Cloud Composer that created and destroyed the dataproc (pyspark) cluster using an airflow scheduler.



Name *
pyspark-pipeline

Node configuration

The configuration information for the Google Kubernetes Engine nodes running the Airflow software.

Node count *
3

The number of nodes in the Google Kubernetes Engine cluster that will be used to run this environment.

Location *
us-west2

The Google Compute Engine region where the environment will be created.

Zone
us-west2-b

The Compute Engine zone in which to deploy the VMs used to run the Apache Airflow software. If unspecified, the service will pick a zone in the Compute Engine region corresponding to the selected location. Must specify location before selecting zone. Learn more.

Machine type
n1-standard-1

The Google Compute Engine machine type used for cluster instances. If unspecified, the machine type will default to 'n1-standard-1'. Must specify location before selecting machine type. Learn more.

Disk size (GB)
20

The disk size in GB used for node VMs. Minimum is 20 GB. If unspecified, defaults to 100 GB. Cannot be updated.

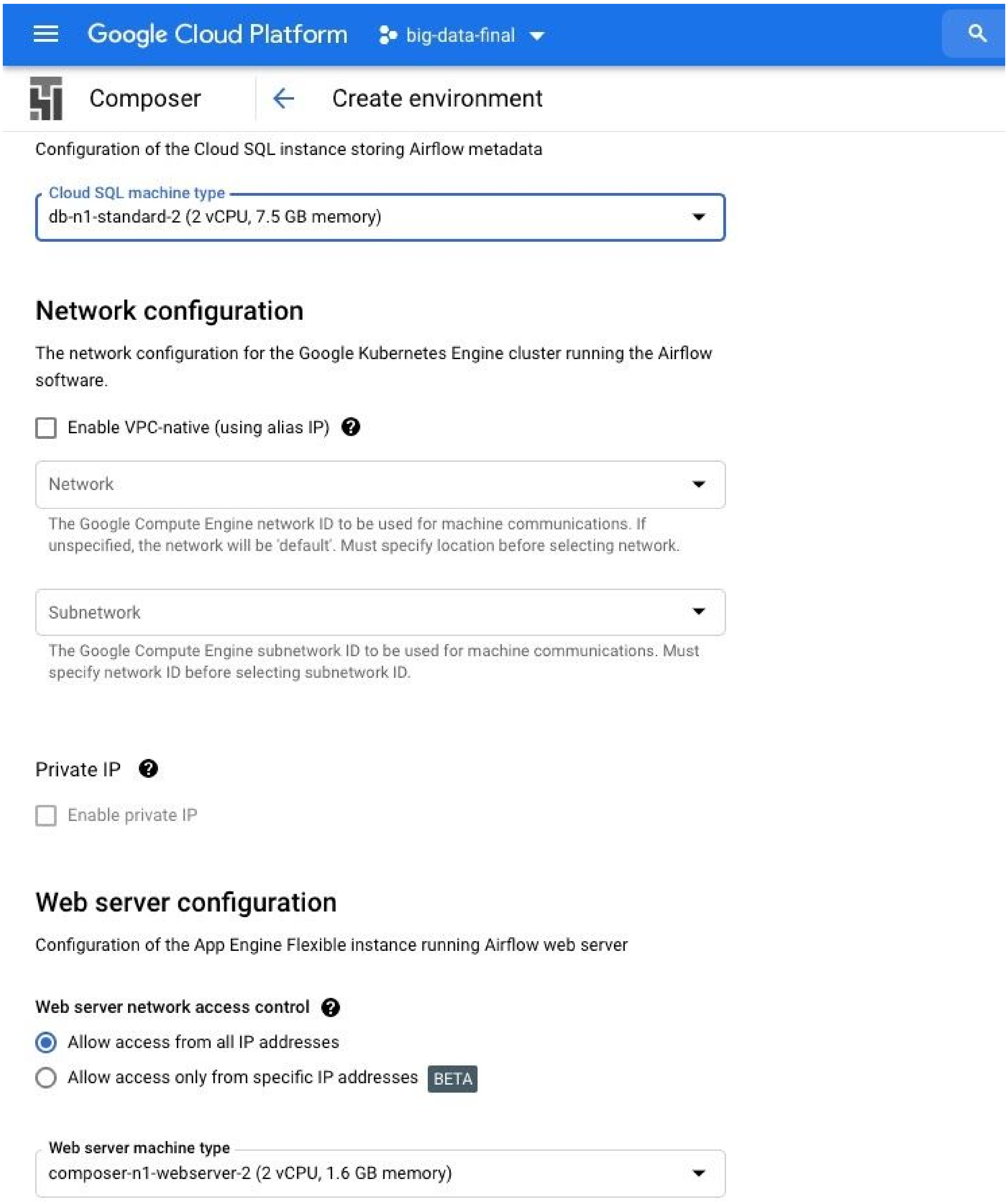
OAuth Scopes

The set of Google API scopes to be made available on all node VMs. If empty, defaults to https://www.googleapis.com/auth/cloud-platform. Cannot be updated. Learn more.

Service account
582004874449-compute@developer.gserviceaccount.com

The Google Cloud Platform Service Account to be used by the node VMs. If a service account is not specified, the "default" Compute Engine service account is used. Cannot be updated.

- Next, we set up a kubernetes cluster using n1-standard v2 machines for master and 3 n1-standard v2 machines with a web server using n1-standard v2 machine to host Airflow GUI web page.



Configuration of the Cloud SQL instance storing Airflow metadata

Cloud SQL machine type: db-n1-standard-2 (2 vCPU, 7.5 GB memory)

Network configuration

The network configuration for the Google Kubernetes Engine cluster running the Airflow software.

☐ Enable VPC-native (using alias IP)

Network

The Google Compute Engine network ID to be used for machine communications. If unspecified, the network will be 'default'. Must specify location before selecting network.

Subnetwork

The Google Compute Engine subnetwork ID to be used for machine communications. Must specify network ID before selecting subnetwork ID.

Private IP

☐ Enable private IP

Web server configuration

Configuration of the App Engine Flexible instance running Airflow web server

Web server network access control

[✓ Allow access from all IP addresses]

Allow access only from specific IP addresses BETA

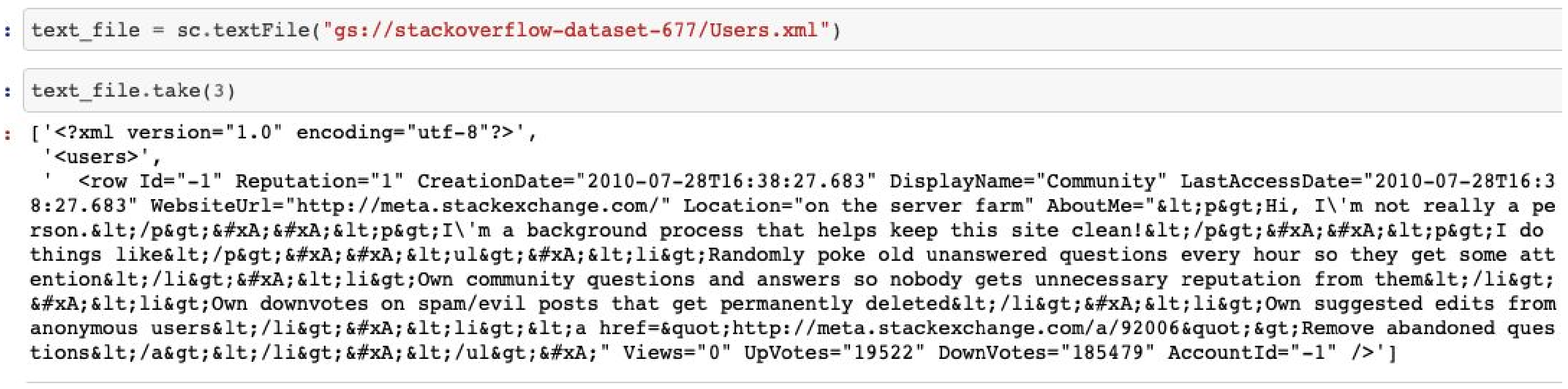
Web server machine type: composer-n1-webserver-2 (2 vCPU, 1.6 GB memory)

Data Processing

To create a dataframe from the xml files, we are using python's **xml.etree.ElementTree** module.

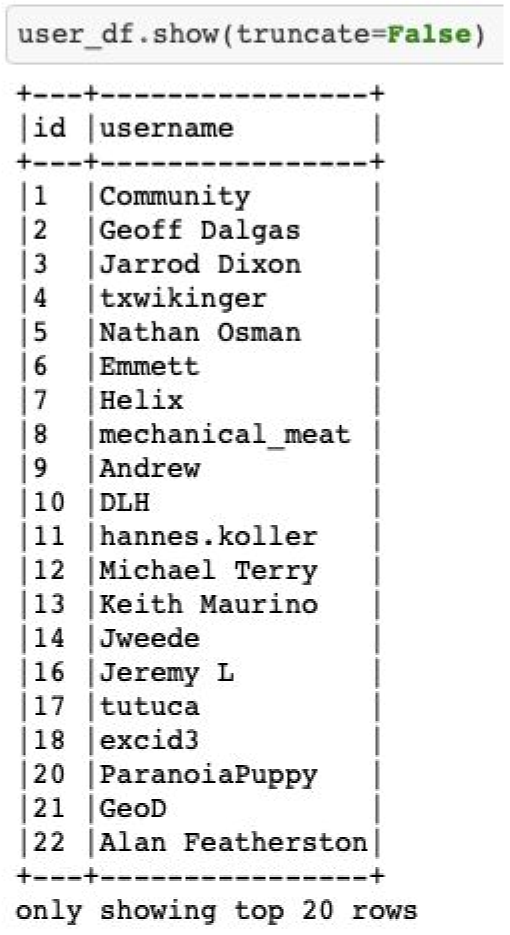
1a) Extracting user id and username and storing to the bucket: jupyter notebook link

Raw Data:



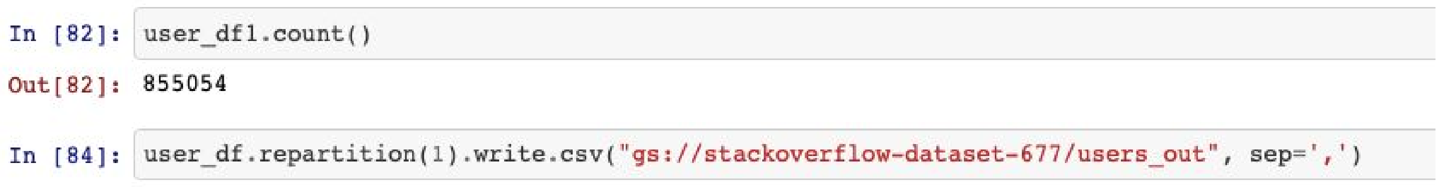
```
: text_file = sc.textFile("gs://stackoverflow-dataset-677/Users.xml")

: text_file.take(3)

: ['<?xml version="1.0" encoding="utf-8"?>',
   '<users>',
   '  <row Id="-1" Reputation="1" CreationDate="2010-07-28T16:38:27.683" DisplayName="Community" LastAccessDate="2010-07-28T16:3
  8:27.683" WebsiteUrl="http://meta.stackexchange.com/" Location="on the server farm" AboutMe="&lt;p&gt;Hi, I\'m not really a pe
  rson.&lt;/p&gt;&#xA;&#xA;&lt;p&gt;I\'m a background process that helps keep this site clean!&lt;/p&gt;&#xA;&#xA;&lt;p&gt;I do
  things like&lt;/p&gt;&#xA;&#xA;&lt;ul&gt;&#xA;&lt;li&gt;Randomly poke old unanswered questions every hour so they get some att
  ention&lt;/li&gt;&#xA;&lt;li&gt;Own community questions and answers so nobody gets unnecessary reputation from them&lt;/li&gt;
  &#xA;&lt;li&gt;Own downvotes on spam/evil posts that get permanently deleted&lt;/li&gt;&#xA;&lt;li&gt;Own suggested edits from
  anonymous users&lt;/li&gt;&#xA;&lt;li&gt;&lt;a href=&quot;http://meta.stackexchange.com/a/92006&quot;&gt;Remove abandoned ques
  tions&lt;/a&gt;&lt;/li&gt;&#xA;&lt;/ul&gt;&#xA;" Views="0" UpVotes="19522" DownVotes="185479" AccountId="-1" />']
```

- After cleaning, filtering and converting the data to dataframe, we get:



```
user_df.show(truncate=False)

+---+----------------+
|id |username        |
+---+----------------+
|1  |Community       |
|2  |Geoff Dalgas    |
|3  |Jarrod Dixon    |
|4  |txwikinger      |
|5  |Nathan Osman    |
|6  |Emmett          |
|7  |Helix           |
|8  |mechanical_meat |
|9  |Andrew          |
|10 |DLH             |
|11 |hannes.koller   |
|12 |Michael Terry   |
|13 |Keith Maurino   |
|14 |Jweede          |
|16 |Jeremy L        |
|17 |tutuca          |
|18 |excid3          |
|20 |ParanoiaPuppy   |
|21 |GeoD            |
|22 |Alan Featherston|
+---+----------------+
only showing top 20 rows
```

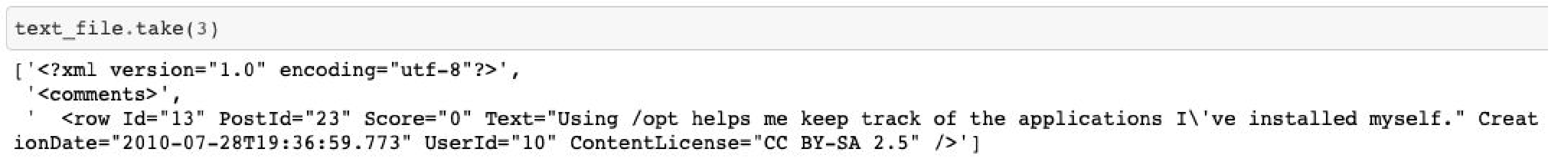
- On completion of creating the dataframe, we store the data to GCS



```
In [82]: user_df1.count()

Out[82]: 855054

In [84]: user_df.repartition(1).write.csv("gs://stackoverflow-dataset-677/users_out", sep=',')
```

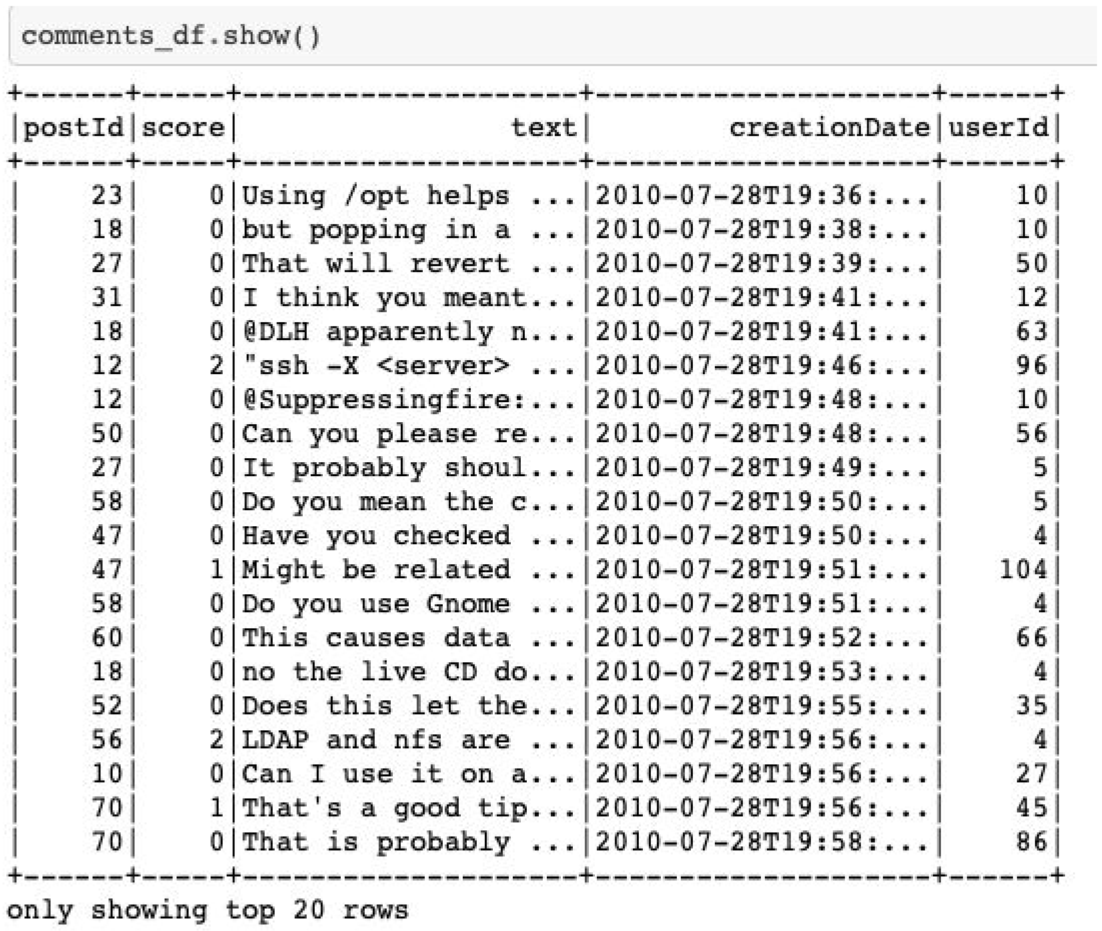
1b) Extracting the comments and joining username with Spark SQL: jupyter notebook link

Raw Data:



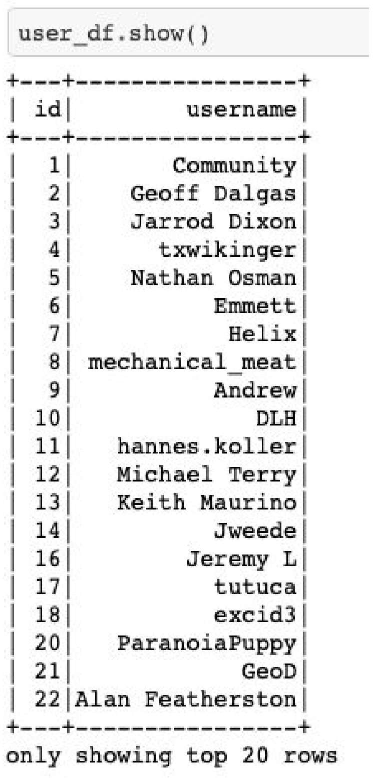
```
text_file.take(3)

['<?xml version="1.0" encoding="utf-8"?>',
 '<comments>',
 '  <row Id="13" PostId="23" Score="0" Text="Using /opt helps me keep track of the applications I\'ve installed myself." Creat
ionDate="2010-07-28T19:36:59.773" UserId="10" ContentLicense="CC BY-SA 2.5" />']
```

- After cleaning, filtering and converting the data to dataframe, we get:



```
comments_df.show()

+------+-----+--------------------+--------------------+------+
|postId|score|                text|        creationDate|userId|
+------+-----+--------------------+--------------------+------+
|    23|    0|Using /opt helps ...|2010-07-28T19:36:...|    10|
|    18|    0|but popping in a ...|2010-07-28T19:38:...|    10|
|    27|    0|That will revert ...|2010-07-28T19:39:...|    50|
|    31|    0|I think you meant...|2010-07-28T19:41:...|    12|
|    18|    0|@DLH apparently n...|2010-07-28T19:41:...|    63|
|    12|    2|"ssh -X <server> ...|2010-07-28T19:46:...|    96|
|    12|    0|@Suppressingfire:...|2010-07-28T19:48:...|    10|
|    50|    0|Can you please re...|2010-07-28T19:48:...|    56|
|    27|    0|It probably shoul...|2010-07-28T19:49:...|     5|
|    58|    0|Do you mean the c...|2010-07-28T19:50:...|     5|
|    47|    0|Have you checked ...|2010-07-28T19:50:...|     4|
|    47|    1|Might be related ...|2010-07-28T19:51:...|   104|
|    58|    0|Do you use Gnome ...|2010-07-28T19:51:...|     4|
|    60|    0|This causes data ...|2010-07-28T19:52:...|    66|
|    18|    0|no the live CD do...|2010-07-28T19:53:...|     4|
|    52|    0|Does this let the...|2010-07-28T19:55:...|    35|
|    56|    2|LDAP and nfs are ...|2010-07-28T19:56:...|     4|
|    10|    0|Can I use it on a...|2010-07-28T19:56:...|    27|
|    70|    1|That's a good tip...|2010-07-28T19:56:...|    45|
|    70|    0|That is probably ...|2010-07-28T19:58:...|    86|
+------+-----+--------------------+--------------------+------+
only showing top 20 rows
```

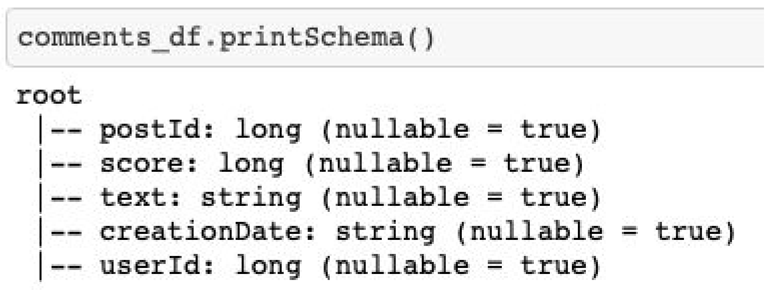
- Next, we read the username and user id from the csv file created in the previous step.



```
user_df.show()
```

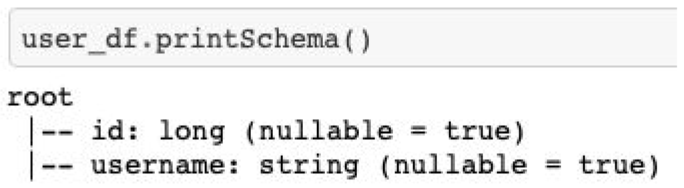
```
+---+----------------+
| id|        username|
+---+----------------+
|  1|       Community|
|  2|    Geoff Dalgas|
|  3|    Jarrod Dixon|
|  4|      txwikinger|
|  5|    Nathan Osman|
|  6|          Emmett|
|  7|           Helix|
|  8| mechanical_meat|
|  9|          Andrew|
| 10|             DLH|
| 11|   hannes.koller|
| 12|   Michael Terry|
| 13|   Keith Maurino|
| 14|          Jweede|
| 16|        Jeremy L|
| 17|          tutuca|
| 18|          excid3|
| 20|   ParanoiaPuppy|
| 21|            GeoD|
| 22|Alan Featherston|
+---+----------------+
only showing top 20 rows
```

- To allow join queries, we made the columns to be integer/long type.



```
comments_df.printSchema()
```

```
root
 |-- postId: long (nullable = true)
 |-- score: long (nullable = true)
 |-- text: string (nullable = true)
 |-- creationDate: string (nullable = true)
 |-- userId: long (nullable = true)
```



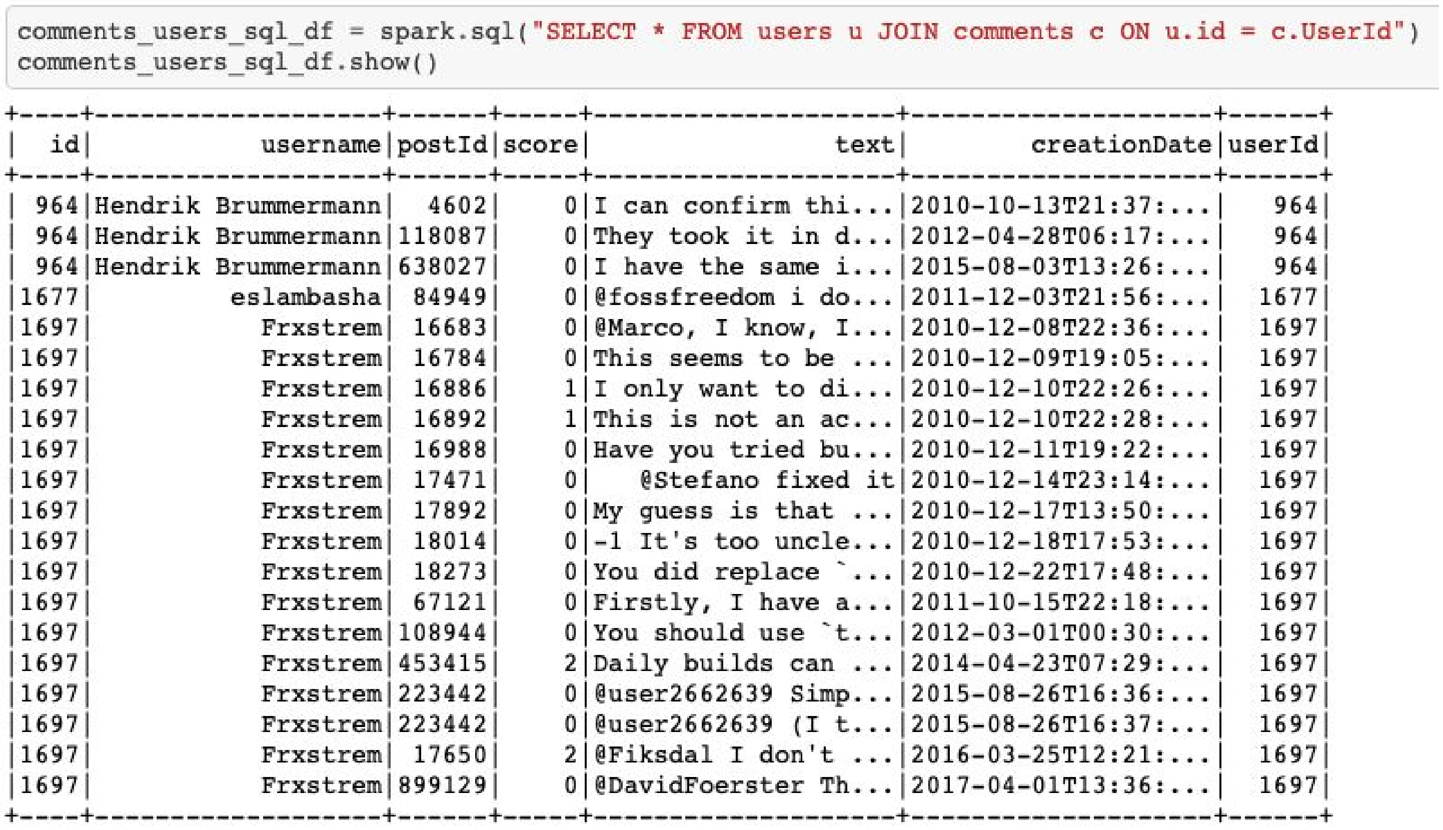
```
user_df.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- username: string (nullable = true)
```

- Converting both datasets to TempView for allowing SQL queries.

```
comments_df.createOrReplaceTempView("comments")
user_df.createOrReplaceTempView("users")
```

- Inner Join Query, introducing username to comments dataframe.

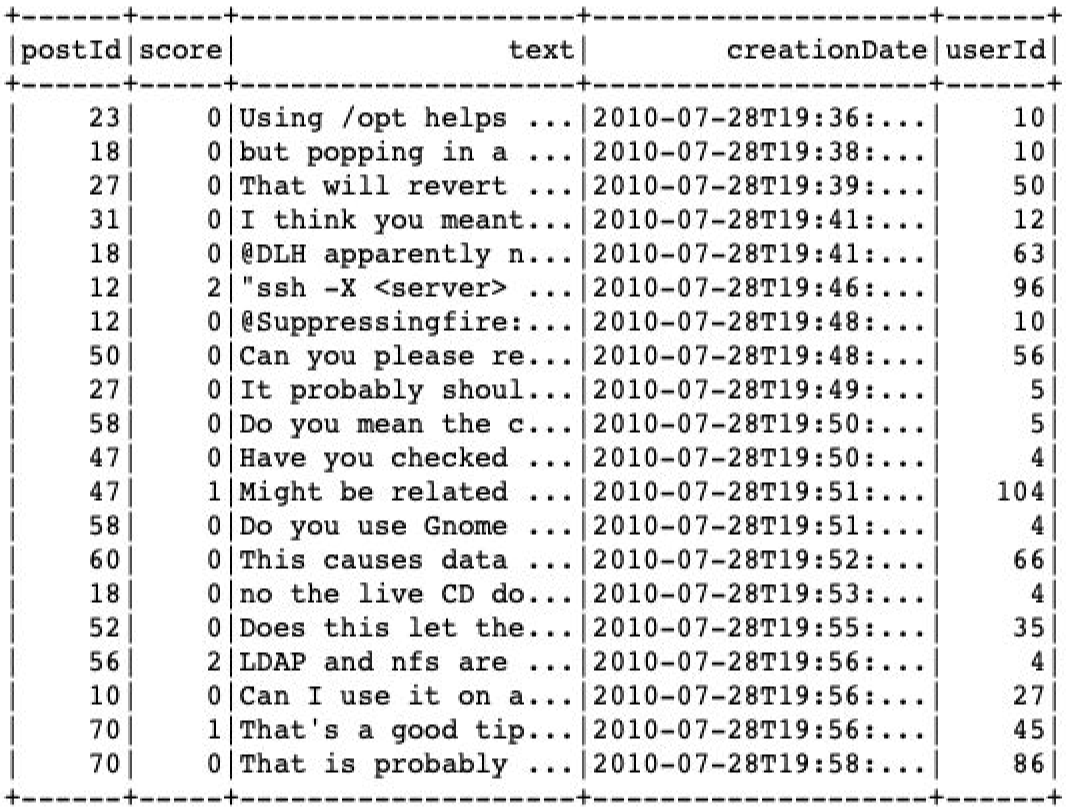


```
comments_users_sql_df = spark.sql("SELECT * FROM users u JOIN comments c ON u.id = c.UserId")
comments_users_sql_df.show()
```

```
+----+-------------------+------+-----+--------------------+--------------------+------+
|  id|           username|postId|score|                text|        creationDate|userId|
+----+-------------------+------+-----+--------------------+--------------------+------+
| 964|Hendrik Brummermann|  4602|    0|I can confirm thi...|2010-10-13T21:37:...|   964|
| 964|Hendrik Brummermann|118087|    0|They took it in d...|2012-04-28T06:17:...|   964|
| 964|Hendrik Brummermann|638027|    0|I have the same i...|2015-08-03T13:26:...|   964|
|1677|         eslambasha| 84949|    0|@fossfreedom i do...|2011-12-03T21:56:...|  1677|
|1697|           Frxstrem| 16683|    0|@Marco, I know, I...|2010-12-08T22:36:...|  1697|
|1697|           Frxstrem| 16784|    0|This seems to be ...|2010-12-09T19:05:...|  1697|
|1697|           Frxstrem| 16886|    1|I only want to di...|2010-12-10T22:26:...|  1697|
|1697|           Frxstrem| 16892|    1|This is not an ac...|2010-12-10T22:28:...|  1697|
|1697|           Frxstrem| 16988|    0|Have you tried bu...|2010-12-11T19:22:...|  1697|
|1697|           Frxstrem| 17471|    0|   @Stefano fixed it|2010-12-14T23:14:...|  1697|
|1697|           Frxstrem| 17892|    0|My guess is that ...|2010-12-17T13:50:...|  1697|
|1697|           Frxstrem| 18014|    0|-1 It's too uncle...|2010-12-18T17:53:...|  1697|
|1697|           Frxstrem| 18273|    0|You did replace `...|2010-12-22T17:48:...|  1697|
|1697|           Frxstrem| 67121|    0|Firstly, I have a...|2011-10-15T22:18:...|  1697|
|1697|           Frxstrem|108944|    0|You should use `t...|2012-03-01T00:30:...|  1697|
|1697|           Frxstrem|453415|    2|Daily builds can ...|2014-04-23T07:29:...|  1697|
|1697|           Frxstrem|223442|    0|@user2662639 Simp...|2015-08-26T16:36:...|  1697|
|1697|           Frxstrem|223442|    0|@user2662639 (I t...|2015-08-26T16:37:...|  1697|
|1697|           Frxstrem| 17650|    2|@Fiksdal I don't ...|2016-03-25T12:21:...|  1697|
|1697|           Frxstrem|899129|    0|@DavidFoerster Th...|2017-04-01T13:36:...|  1697|
+----+-------------------+------+-----+--------------------+--------------------+------+
```

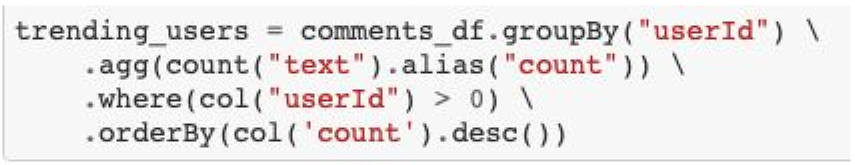
2. Trending users with max comments:

- After extracting the comments data and converting to dataframe, we got:



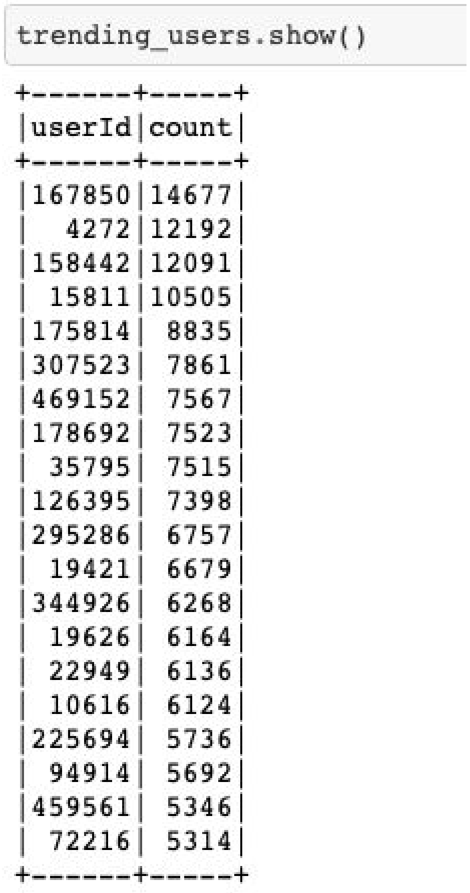
```
+------+-----+--------------------+--------------------+------+
|postId|score|                text|        creationDate|userId|
+------+-----+--------------------+--------------------+------+
|    23|    0|Using /opt helps ...|2010-07-28T19:36:...|    10|
|    18|    0|but popping in a ...|2010-07-28T19:38:...|    10|
|    27|    0|That will revert ...|2010-07-28T19:39:...|    50|
|    31|    0|I think you meant...|2010-07-28T19:41:...|    12|
|    18|    0|@DLH apparently n...|2010-07-28T19:41:...|    63|
|    12|    2|"ssh -X <server> ...|2010-07-28T19:46:...|    96|
|    12|    0|@Suppressingfire:...|2010-07-28T19:48:...|    10|
|    50|    0|Can you please re...|2010-07-28T19:48:...|    56|
|    27|    0|It probably shoul...|2010-07-28T19:49:...|     5|
|    58|    0|Do you mean the c...|2010-07-28T19:50:...|     5|
|    47|    0|Have you checked ...|2010-07-28T19:50:...|     4|
|    47|    1|Might be related ...|2010-07-28T19:51:...|   104|
|    58|    0|Do you use Gnome ...|2010-07-28T19:51:...|     4|
|    60|    0|This causes data ...|2010-07-28T19:52:...|    66|
|    18|    0|no the live CD do...|2010-07-28T19:53:...|     4|
|    52|    0|Does this let the...|2010-07-28T19:55:...|    35|
|    56|    2|LDAP and nfs are ...|2010-07-28T19:56:...|     4|
|    10|    0|Can I use it on a...|2010-07-28T19:56:...|    27|
|    70|    1|That's a good tip...|2010-07-28T19:56:...|    45|
|    70|    0|That is probably ...|2010-07-28T19:58:...|    86|
+------+-----+--------------------+--------------------+------+
```

- Next, we find the trending users by using groupBy aggregate function and sorting it with respect to count in descending order.



```
trending_users = comments_df.groupBy("userId") \
     .agg(count("text").alias("count")) \
     .where(col("userId") > 0) \
     .orderBy(col('count').desc())
```

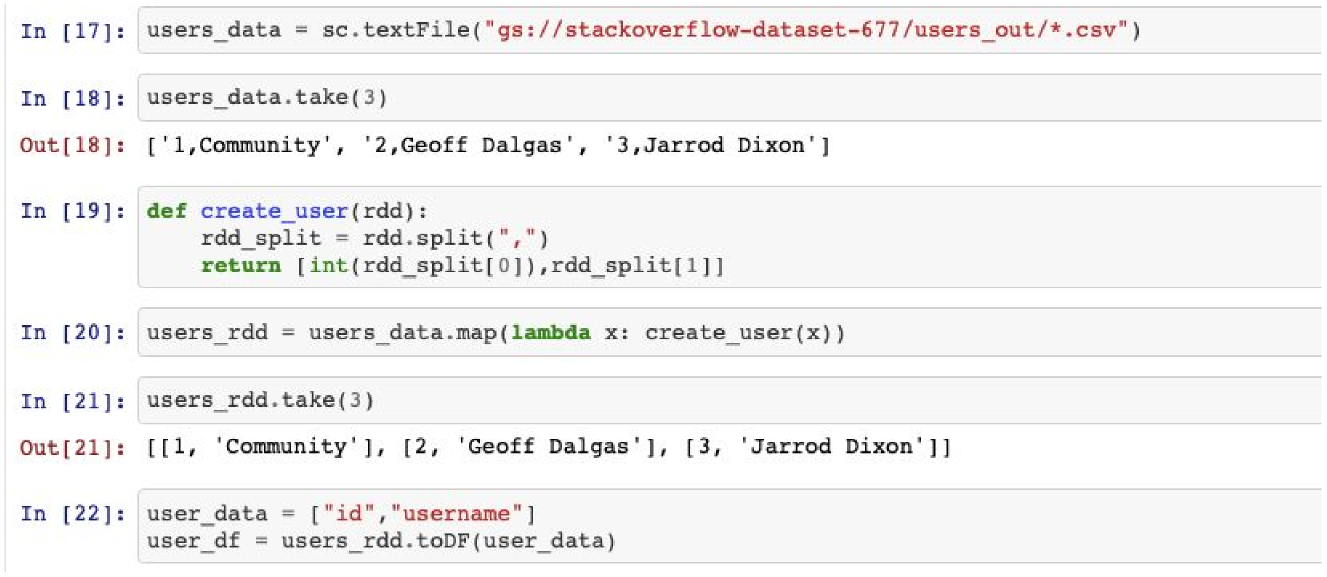
- Result



```
trending_users.show()
```

```
+------+-----+
|userId|count|
+------+-----+
|167850|14677|
|  4272|12192|
|158442|12091|
| 15811|10505|
|175814| 8835|
|307523| 7861|
|469152| 7567|
|178692| 7523|
| 35795| 7515|
|126395| 7398|
|295286| 6757|
| 19421| 6679|
|344926| 6268|
| 19626| 6164|
| 22949| 6136|
| 10616| 6124|
|225694| 5736|
| 94914| 5692|
|459561| 5346|
| 72216| 5314|
+------+-----+
```

- Next, we create a temp view of the trending users and merge it with usernames obtained using 1a.



```
In [17]: users_data = sc.textFile("gs://stackoverflow-dataset-677/users_out/*.csv")

In [18]: users_data.take(3)

Out[18]: ['1,Community', '2,Geoff Dalgas', '3,Jarrod Dixon']

In [19]: def create_user(rdd):
             rdd_split = rdd.split(",")
             return [int(rdd_split[0]),rdd_split[1]]

In [20]: users_rdd = users_data.map(lambda x: create_user(x))

In [21]: users_rdd.take(3)

Out[21]: [[1, 'Community'], [2, 'Geoff Dalgas'], [3, 'Jarrod Dixon']]

In [22]: user_data = ["id","username"]
         user_df = users_rdd.toDF(user_data)
```

- Joining the usernames with the trending user.

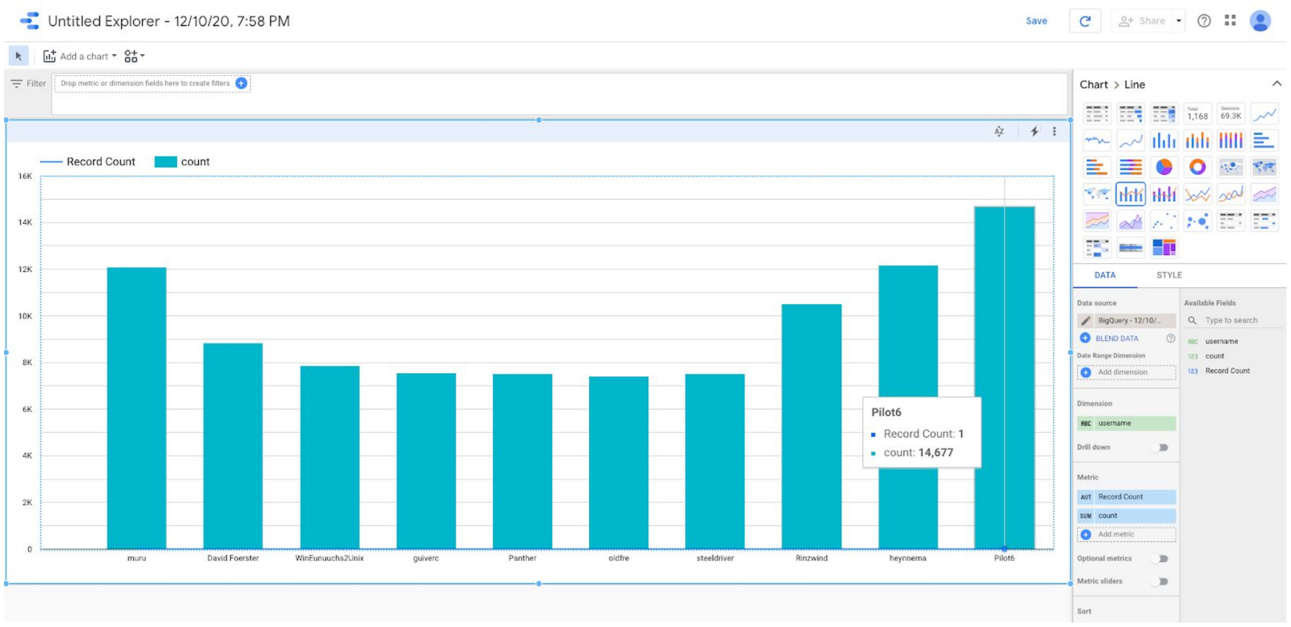
```
trending_usernames = spark.sql("SELECT count,username FROM trending_users tu JOIN users u ON u.id = tu.userId order by count d
esc limit 10")
trending_usernames.show()
```

```
+-----+----------------+
|count|        username|
+-----+----------------+
|14677|          Pilot6|
|12192|        heynnema|
|12091|            muru|
|10505|        Rinzwind|
| 8835|  David Foerster|
| 7861|WinEunuuchs2Unix|
| 7567|         guiverc|
| 7523|     steeldriver|
| 7515|         Panther|
| 7398|         oldfred|
+-----+----------------+
```

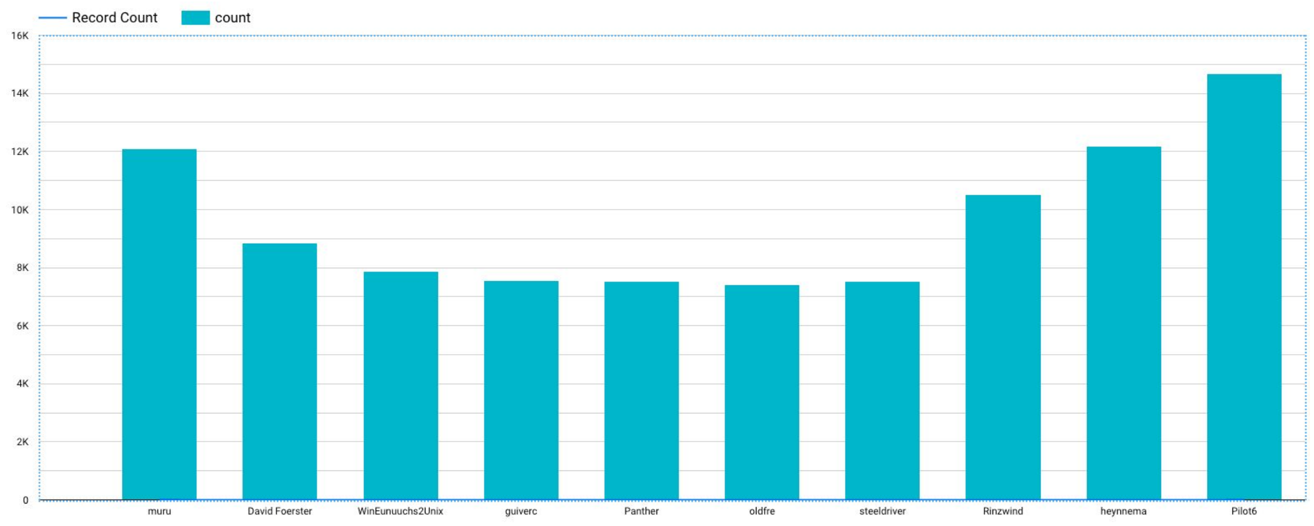
- Storing it to GCS, will be imported to Google BigQuery for visualization with Google Data Studio.

```
trending_usernames.repartition(1).write.csv("gs://stackoverflow-dataset-677/trending_usernames", sep=',')
```

- After importing the dataset to bigquery, and exporting the data to Google Data Studio for visualization.



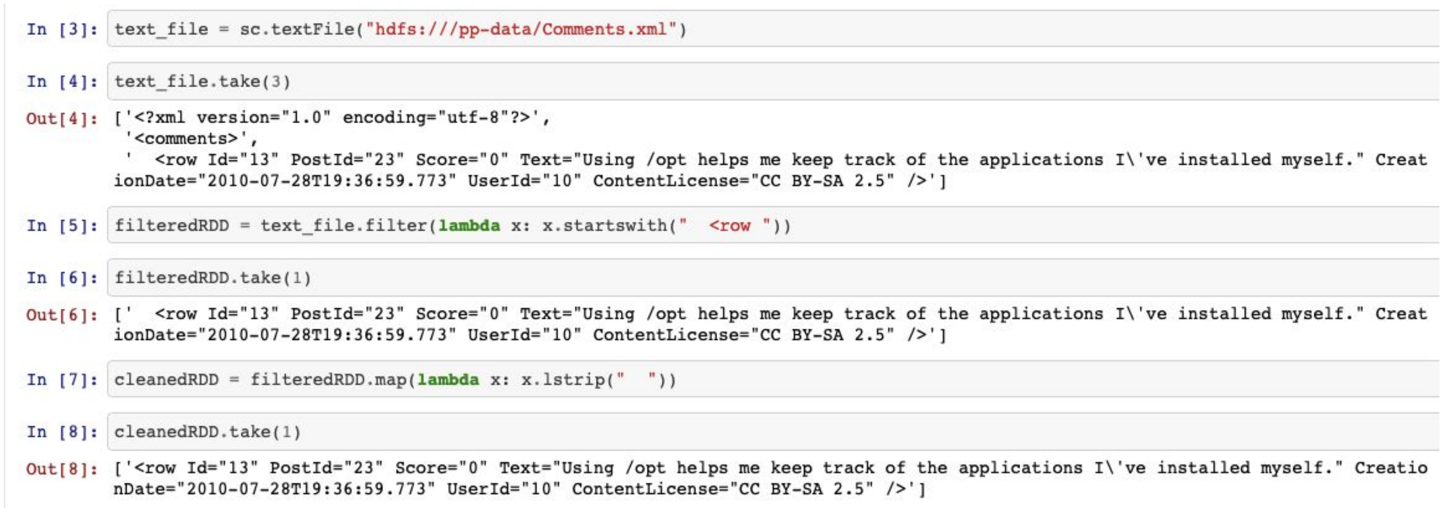
- Result



3: Trending Topics in Comments:

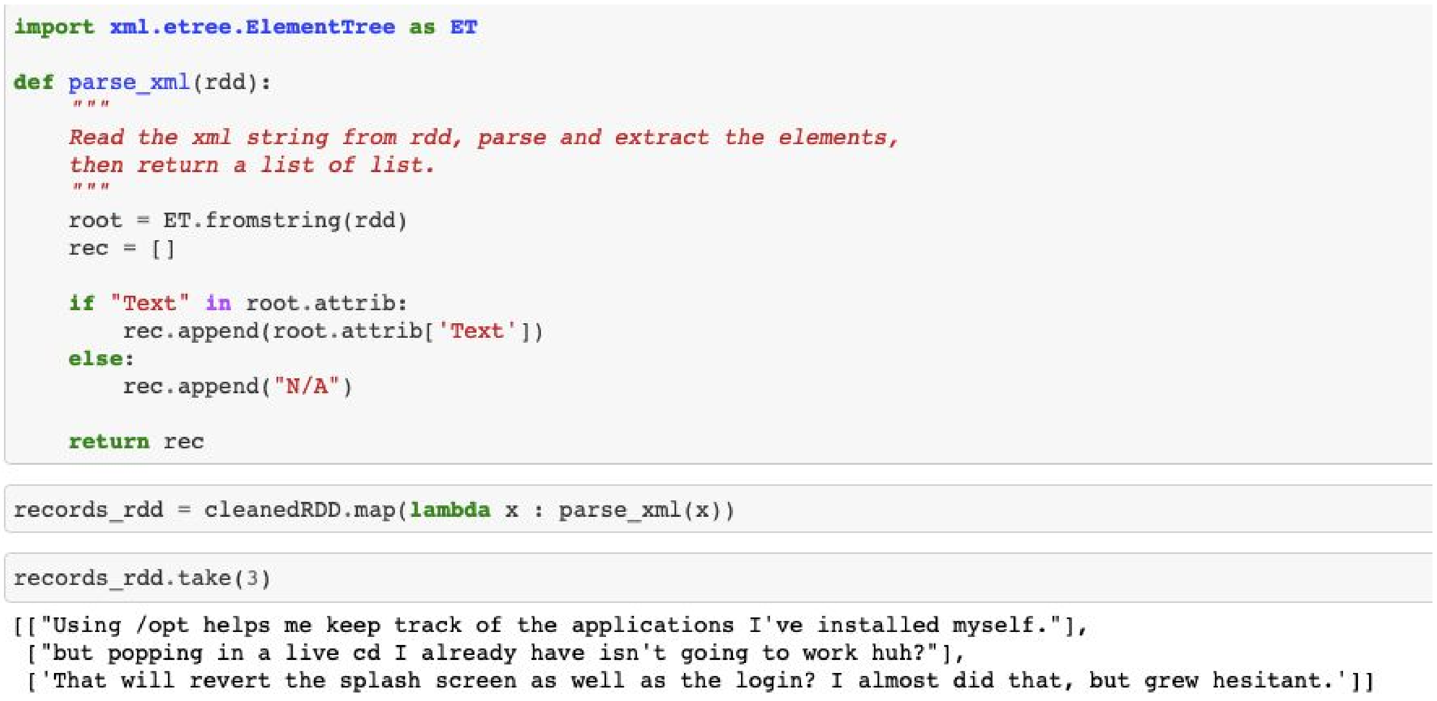
Notebook Link:

- Obtaining the data from GCS and cleaning it.



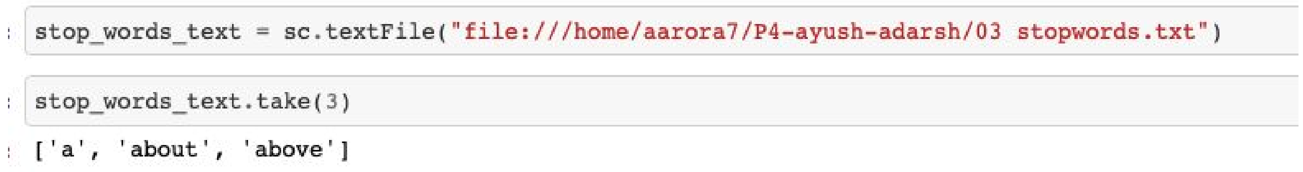
```
In [3]: text_file = sc.textFile("hdfs:///pp-data/Comments.xml")

In [4]: text_file.take(3)

Out[4]: ['<?xml version="1.0" encoding="utf-8"?>',
         '<comments>',
         '  <row Id="13" PostId="23" Score="0" Text="Using /opt helps me keep track of the applications I\'ve installed myself." Creat
        ionDate="2010-07-28T19:36:59.773" UserId="10" ContentLicense="CC BY-SA 2.5" />']

In [5]: filteredRDD = text_file.filter(lambda x: x.startswith("  <row "))

In [6]: filteredRDD.take(1)

Out[6]: ['  <row Id="13" PostId="23" Score="0" Text="Using /opt helps me keep track of the applications I\'ve installed myself." Creat
        ionDate="2010-07-28T19:36:59.773" UserId="10" ContentLicense="CC BY-SA 2.5" />']

In [7]: cleanedRDD = filteredRDD.map(lambda x: x.lstrip("  "))

In [8]: cleanedRDD.take(1)

Out[8]: ['<row Id="13" PostId="23" Score="0" Text="Using /opt helps me keep track of the applications I\'ve installed myself." Creatio
        nDate="2010-07-28T19:36:59.773" UserId="10" ContentLicense="CC BY-SA 2.5" />']
```

- After extracting the attributes (comment text) from xml,we get



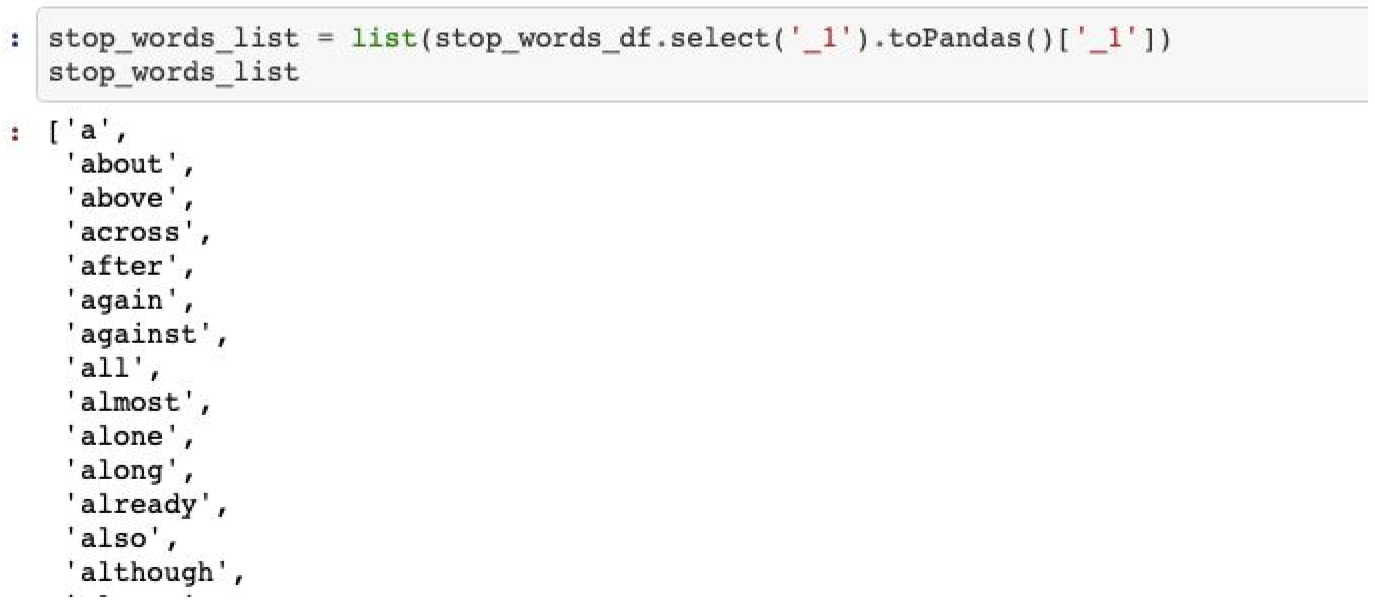
```
import xml.etree.ElementTree as ET

def parse_xml(rdd):
    """
    Read the xml string from rdd, parse and extract the elements,
    then return a list of list.
    """
    root = ET.fromstring(rdd)
    rec = []

    if "Text" in root.attrib:
        rec.append(root.attrib['Text'])
    else:
        rec.append("N/A")

    return rec

records_rdd = cleanedRDD.map(lambda x : parse_xml(x))

records_rdd.take(3)

[["Using /opt helps me keep track of the applications I've installed myself."],
 ["but popping in a live cd I already have isn't going to work huh?"],
 ['That will revert the splash screen as well as the login? I almost did that, but grew hesitant.']]
```

- Next, we have to remove the stop words from this text. So, we downloaded the stop words list online. Importing the stop words from file



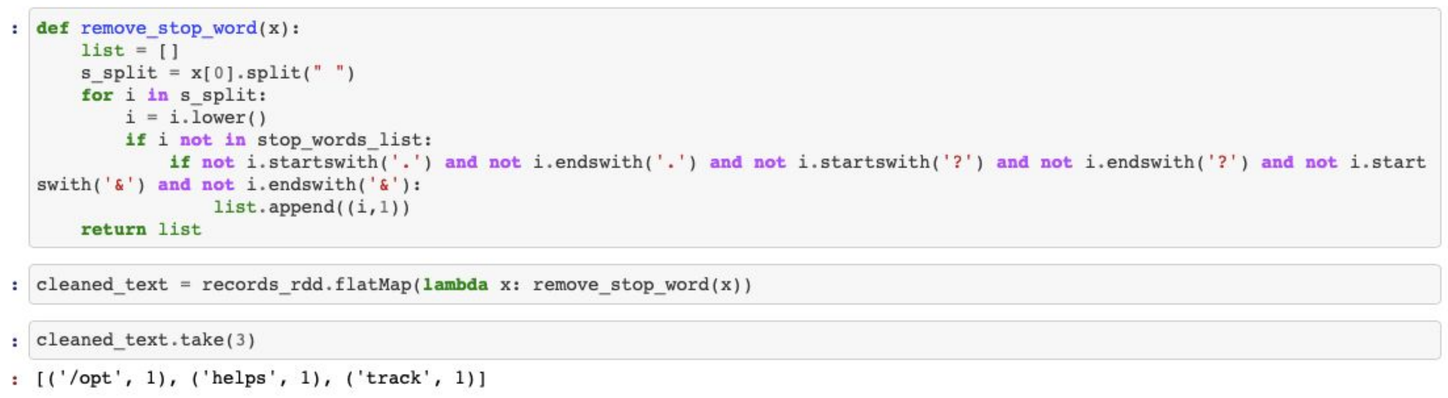
```
stop_words_text = sc.textFile("file:///home/aarora7/P4-ayush-adarsh/03 stopwords.txt")

stop_words_text.take(3)

['a', 'about', 'above']
```

- Next, we converted it into dataframe. Converting the stop words dataframe to list
-



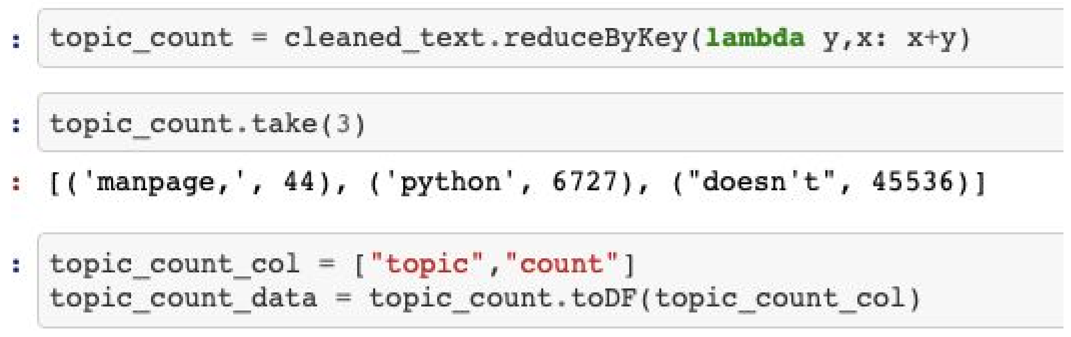
```
: stop_words_list = list(stop_words_df.select('_1').toPandas()['_1'])
  stop_words_list

: ['a',
   'about',
   'above',
   'across',
   'after',
   'again',
   'against',
   'all',
   'almost',
   'alone',
   'along',
   'already',
   'also',
   'although',
```

- Next, removing stopwords from main dataframe and converting into (stop_word,1) tuple



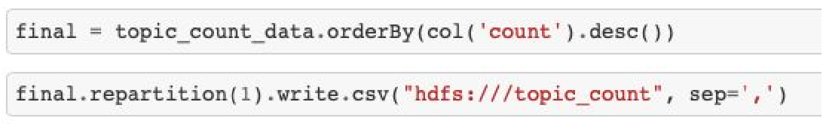
```
: def remove_stop_word(x):
      list = []
      s_split = x[0].split(" ")
      for i in s_split:
          i = i.lower()
          if i not in stop_words_list:
              if not i.startswith('.') and not i.endswith('.') and not i.startswith('?') and not i.endswith('?') and not i.start
swith('&') and not i.endswith('&'):
                  list.append((i,1))
      return list

: cleaned_text = records_rdd.flatMap(lambda x: remove_stop_word(x))

: cleaned_text.take(3)

: [('/opt', 1), ('helps', 1), ('track', 1)]
```

- Next, finding the total count of words by using reduceByKey and converting it to Dataframe for storing to GCS.



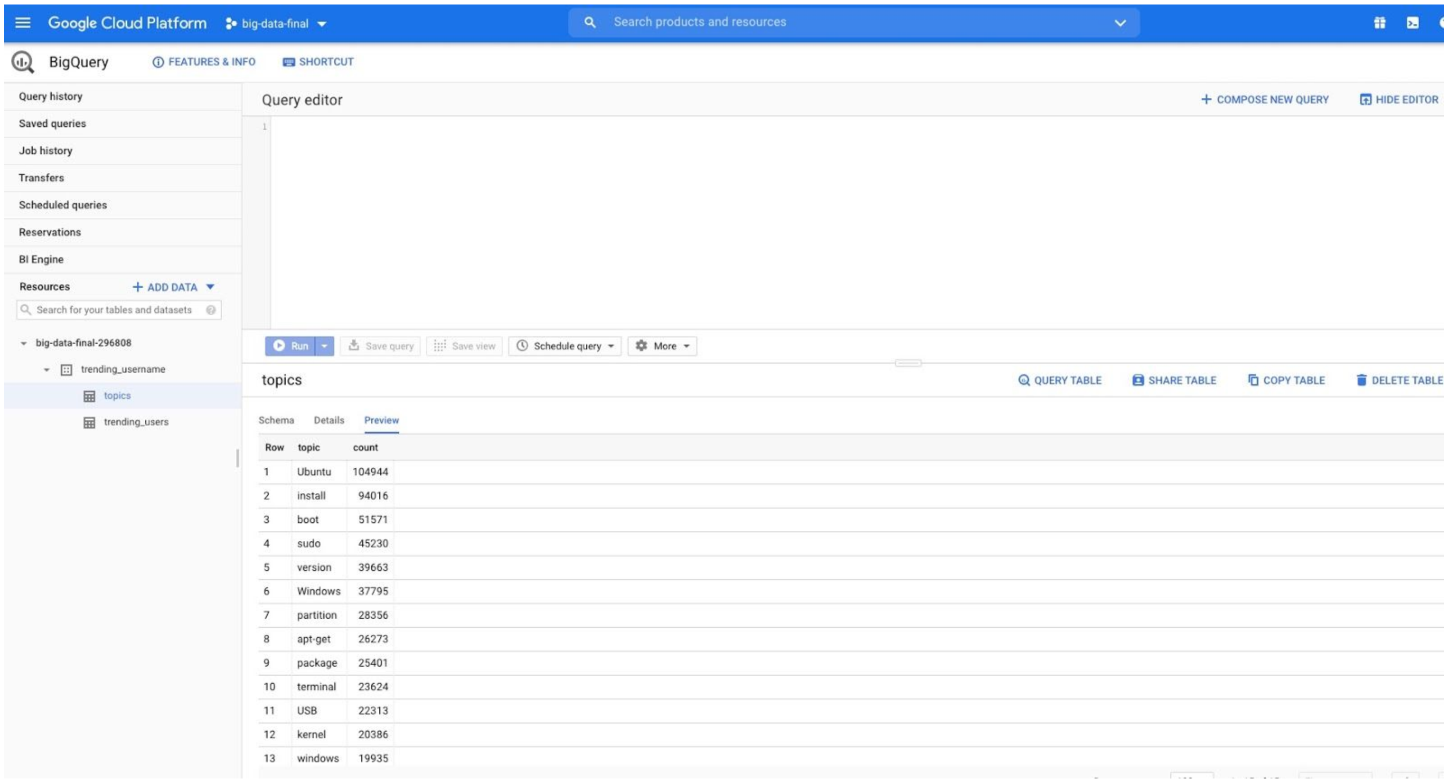
```
: topic_count = cleaned_text.reduceByKey(lambda y,x: x+y)

: topic_count.take(3)

: [('manpage,', 44), ('python', 6727), ("doesn't", 45536)]

: topic_count_col = ["topic","count"]
  topic_count_data = topic_count.toDF(topic_count_col)
```

- Storing to GCS and sorting the DF in descending order w.r.t count



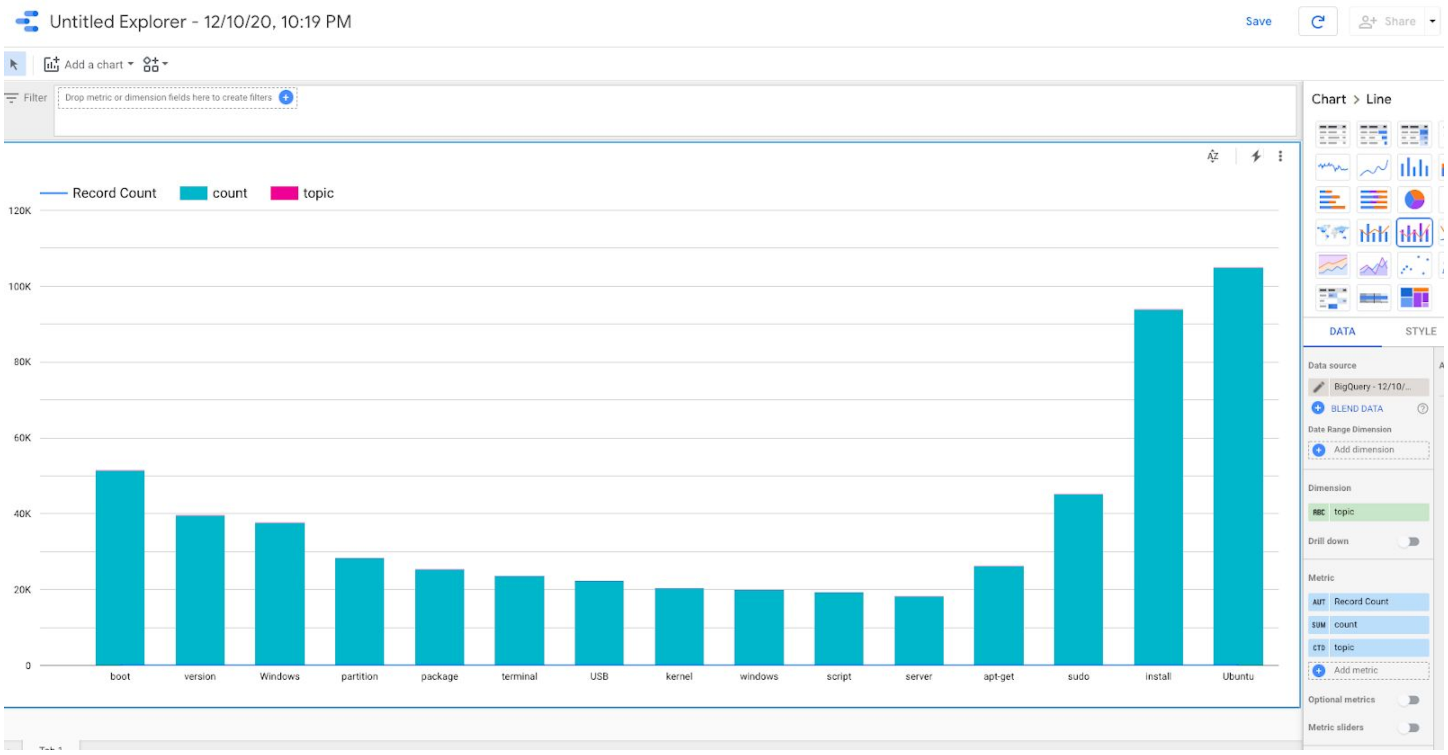
```
final = topic_count_data.orderBy(col('count').desc())

final.repartition(1).write.csv("hdfs:///topic_count", sep=',')
```

- Next, we imported the data to BigQuery from GSC to export to Google Data Studio.

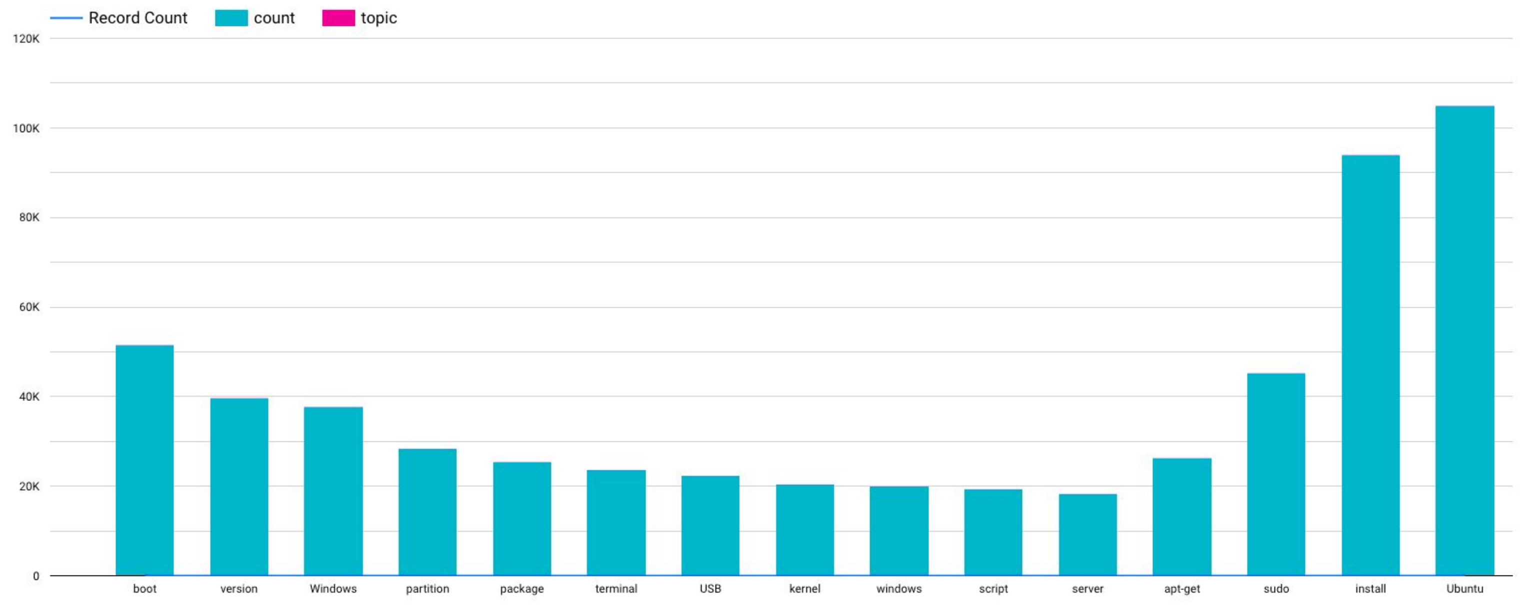


Row	topic	count
1	Ubuntu	104944
2	install	94016
3	boot	51571
4	sudo	45230
5	version	39663
6	Windows	37795
7	partition	28356
8	apt-get	26273
9	package	25401
10	terminal	23624
11	USB	22313
12	kernel	20386
13	windows	19935

- Visualizing the data using Google Data Studio.



- Result: Trending Keywords: Ubuntu is the most trending keyword with install as the second. This means most users comment about installation.



4. Automating Part 1 with Google Cloud Composer (Apache Airflow).

- After setting up the server (shown in the beginning), next we create python scripts for part 1 and 2 and upload them to GCS. The GCS path will be referenced during the final orchestration.

  Part 1: users.py

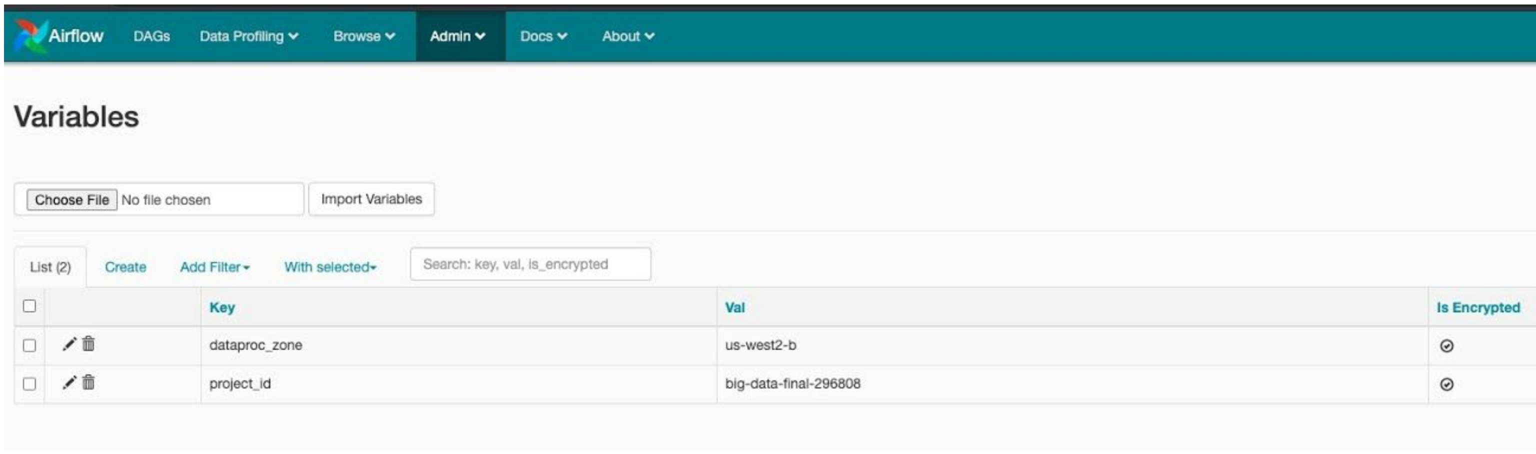
```
from pyspark.sql.functions import *
import time
import pyspark.sql.functions as F
from pyspark.sql.types import *
import xml.etree.ElementTree as ET
from pyspark.sql import SparkSession

spark = SparkSession \
.builder \
.appName("My PySpark code") \
.getOrCreate()

text_file = sc.textFile("gs://stackoverflow-dataset-677/Users.xml")
filteredRDD = text_file.filter(lambda x: x.startswith("  <row "))
cleanedRDD = filteredRDD.map(lambda x: x.lstrip("  "))

def parse_xml(rdd):
   root = ET.fromstring(rdd)
   rec = []
   id = root.attrib['Id']
   if id == "-1":
       id = "1"
   rec.append(id)
   rec.append(root.attrib['DisplayName'])
   return rec

records_rdd = cleanedRDD.map(lambda x : parse_xml(x))

user_data = ["id","username"]
user_df = records_rdd.toDF(user_data)
user_df.repartition(1).write.csv("gs://stackoverflow-dataset-677/users_out1", sep=',')
```

Part 2: comments.py

```
from pyspark.sql.functions import *
import time
import pyspark.sql.functions as F
from pyspark.sql.types import *
import xml.etree.ElementTree as ET
from pyspark.sql import SparkSession

spark = SparkSession \
.builder \
.appName("My PySpark code") \
.getOrCreate()

text_file = sc.textFile("gs://stackoverflow-dataset-677/Comments.xml")
filteredRDD = text_file.filter(lambda x: x.startswith("  <row "))
cleanedRDD = filteredRDD.map(lambda x: x.lstrip("  "))

def parse_xml(rdd):
   """
   Read the xml string from rdd, parse and extract the elements,
   then return a list of list.
   """
   root = ET.fromstring(rdd)
   rec = []

   if "PostId" in root.attrib:
       rec.append(int(root.attrib['PostId']))
   else:
       rec.append(0)
   if "Score" in root.attrib:
       rec.append(int(root.attrib['Score']))
   else:
       rec.append(0)
```

```
    if "Text" in root.attrib:
        rec.append(root.attrib['Text'])
    else:
        rec.append("N/A")
    if "CreationDate" in root.attrib:
        rec.append(root.attrib['CreationDate'])
    else:
        rec.append("N/A")
    if "UserId" in root.attrib:
        rec.append(int(root.attrib['UserId']))
    else:
        rec.append(0)
    return rec

records_rdd = cleanedRDD.map(lambda x : parse_xml(x))
comments_data = ["postId","score","text","creationDate","userId"]
comments_df = records_rdd.toDF(comments_data)
comments_df.createOrReplaceTempView("comments")
comments_sql_df = spark.sql("SELECT * FROM comments")

users_data = sc.textFile("gs://stackoverflow-dataset-677/users_out1/*.csv")

def create_user(rdd):
    rdd_split = rdd.split(",")
    return [int(rdd_split[0]),rdd_split[1]]

users_rdd = users_data.map(lambda x: create_user(x))
user_data = ["id","username"]
user_df = users_rdd.toDF(user_data)
user_df.createOrReplaceTempView("users")
comments_users_sql_df = spark.sql("SELECT * FROM users u JOIN comments c ON u.id = c.UserId")
comments_users_sql_df.repartition(1).write.csv("gs://stackoverflow-dataset-677/users_out1",
sep=',')
```

- Next, we created two variables for the final orchestration. The GCP project id and the zone where the clusters will be created.



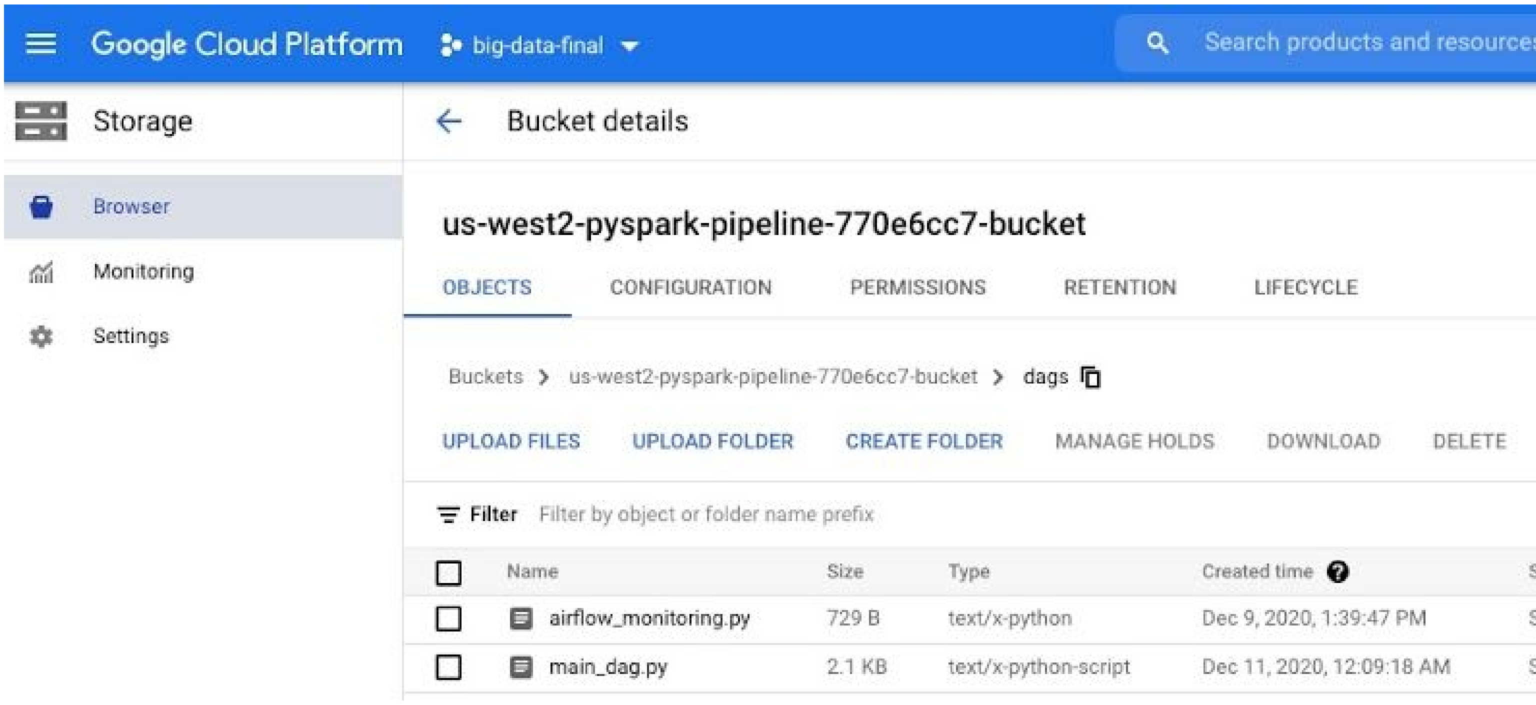
Key	Val	Is Encrypted
dataproc_zone	us-west2-b	✓
project_id	big-data-final-296808	✓

- **Final Orchestration:** To connect both the scripts together and for automatic creation and destruction of Spark Clusters, we created a DAG file.
    - This contains the path of the users.py and comments.py file and the airflow variables as well.

main_dag.py

```
# Ref:
https://medium.com/analytics-vidhya/a-gentle-introduction-to-data-workflows-with-apach
e-airflow-and-apache-spark-6c2cd9aee573
from datetime import timedelta, datetime
from airflow import models
from airflow.operators.bash_operator import BashOperator
from airflow.contrib.operators import dataproc_operator
from airflow.utils import trigger_rule

# STEP 2:Define a start date
#In this case yesterday
yesterday = datetime(2020, 12, 10)

SPARK_CODE = ('gs://stackoverflow-dataset-677/01_user.py')
SPARK_CODE2 = ('gs://stackoverflow-dataset-677/02_user_comments_join.py')
dataproc_job_name = 'extract_users_job_dataproc'
dataproc_job_name2 = 'extract_comments_join_users_dataproc'

# STEP 3: Set default arguments for the DAG
default_dag_args = {
'start_date': yesterday,
'depends_on_past': False,
'email_on_failure': False,
'email_on_retry': False,
'retries': 1,
'retry_delay': timedelta(minutes=5),
'project_id': models.Variable.get('project_id')
}

# STEP 4: Define DAG
# set the DAG name, add a DAG description, define the schedule interval and pass the
default arguments defined before
with models.DAG(
'comments_extract_user_join_spark_workflow',
description='DAG for extracting comments and merging with user name',
schedule_interval=timedelta(days=1),
default_args=default_dag_args) as dag:

# STEP 5: Set Operators
# BashOperator
# A simple print date
   print_date = BashOperator(
   task_id='print_date',
   bash_command='date'
   )

# dataproc_operator
# Create small dataproc cluster
   create_dataproc =  dataproc_operator.DataprocClusterCreateOperator(
   task_id='create_dataproc',
   cluster_name='dataproc-cluster-demo-{{ ds_nodash }}',
   num_workers=2,
   zone=models.Variable.get('dataproc_zone'),
   master_machine_type='n1-standard-1',
   worker_machine_type='n1-standard-1')

   run_spark = dataproc_operator.DataProcPySparkOperator(
   task_id='run_spark',
   main=SPARK_CODE,
   cluster_name='dataproc-cluster-demo-{{ ds_nodash }}',
   job_name=dataproc_job_name
   )

   run_spark2 = dataproc_operator.DataProcPySparkOperator(
   task_id='run_spark2',
   main=SPARK_CODE2,
   cluster_name='dataproc-cluster-demo-{{ ds_nodash }}',
   job_name=dataproc_job_name2
   )

   # dataproc_operator
   # Delete Cloud Dataproc cluster.
   delete_dataproc = dataproc_operator.DataprocClusterDeleteOperator(
   task_id='delete_dataproc',
   cluster_name='dataproc-cluster-demo-{{ ds_nodash }}',
   trigger_rule=trigger_rule.TriggerRule.ALL_DONE)

# STEP 6: Set DAGs dependencies
# Each task should run after have finished the task before.
print_date >> create_dataproc >> run_spark  >> run_spark2  >> delete_dataproc
```

- To run the main dag file, we upload it to DAGs folder and the script ran automatically in 2 minutes.



us-west2-pyspark-pipeline-770e6cc7-bucket

OBJECTS CONFIGURATION PERMISSIONS RETENTION LIFECYCLE

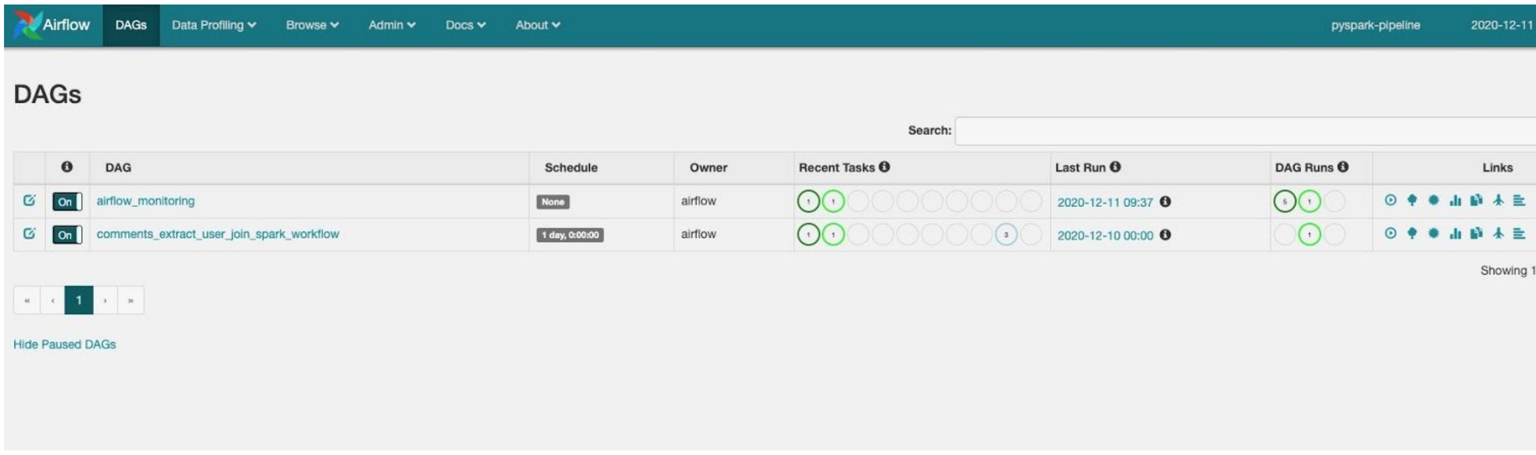
Buckets > us-west2-pyspark-pipeline-770e6cc7-bucket > dags

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter Filter by object or folder name prefix

Name	Size	Type	Created time
airflow_monitoring.py	729 B	text/x-python	Dec 9, 2020, 1:39:47 PM
main_dag.py	2.1 KB	text/x-python-script	Dec 11, 2020, 12:09:18 AM

- Main dag file with with workflow name “Comments_extract_user_join_spark_workflow.



DAGs

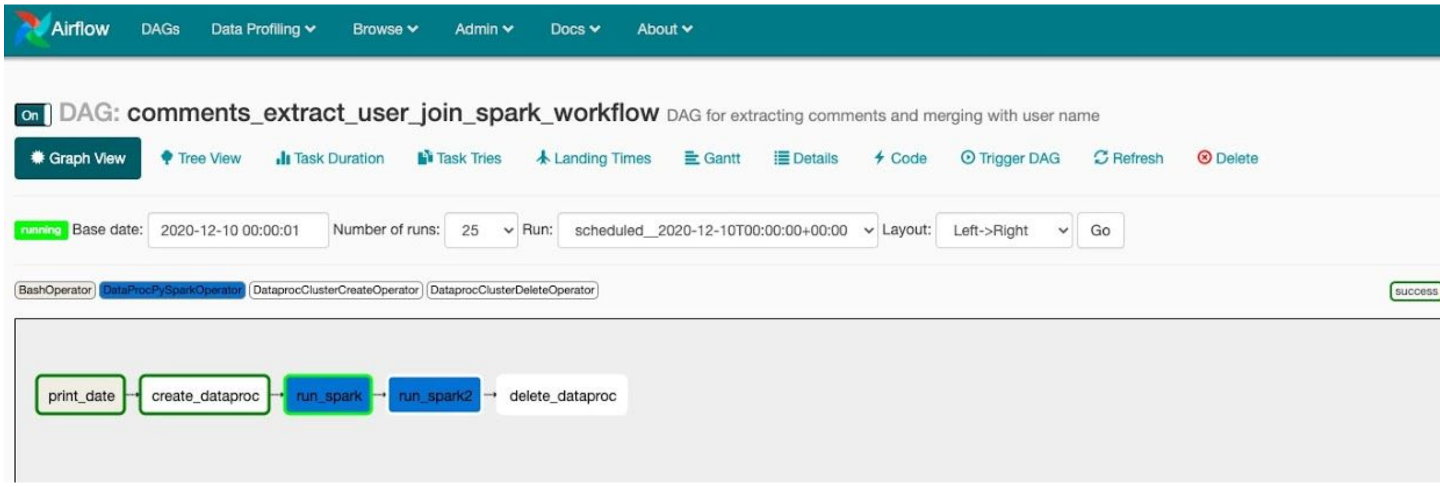
Search:

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
On	airflow_monitoring	None	airflow	1 1	2020-12-11 09:37	5 1	
On	comments_extract_user_join_spark_workflow	1 day, 0:00:00	airflow	1 1 3	2020-12-10 00:00	1	

Showing 1

Hide Paused DAGs

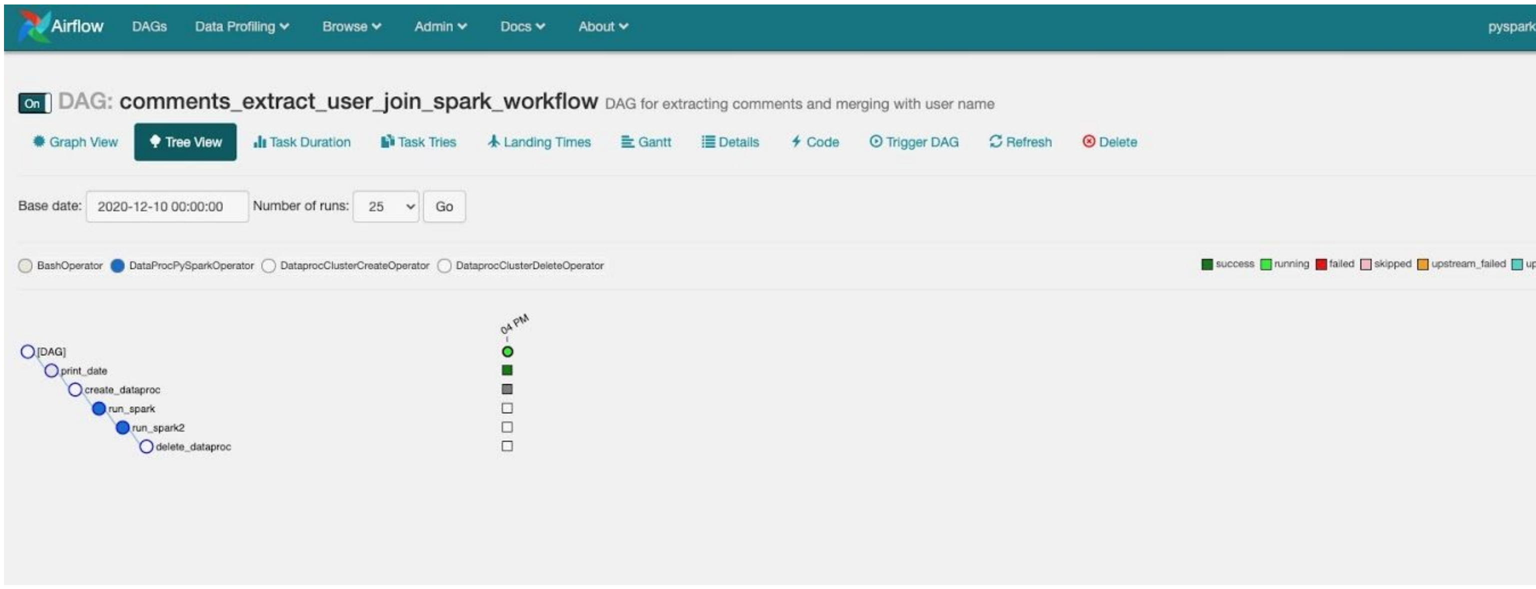
- Graph View of the Workflow



On DAG: comments_extract_user_join_spark_workflow DAG for extracting comments and merging with user name

running Base date: 2020-12-10 00:00:01 Number of runs: 25 Run: scheduled__2020-12-10T00:00:00+00:00 Layout: Left->Right Go

- Tree View



DAG: comments_extract_user_join_spark_workflow DAG for extracting comments and merging with user name

Base date: 2020-12-10 00:00:00 Number of runs: 25

BashOperator ● DataProcPySparkOperator DataprocClusterCreateOperator DataprocClusterDeleteOperator

🟩 success 🟩 running 🟥 failed skipped upstream_failed

[DAG]
print_date
create_dataproc
run_spark
run_spark2
delete_dataproc

- Finally, the output file was saved in GCS.