Using Spark and Watson on IBM Bluemix for Sentiment Analytics

Miracle Summer of Code Virtual Labs Series

Mounika Chirukuri

Lead Researcher – Big Data/Analytics

Miracle Software Systems, Inc.

April 26th, 2016

Miracle's SoC Series : Using Spark and Watson on IBM Bluemix for Sentiment Analytics

Overview

In this lab we will be using Twitter Apps, Spark and Watson from within Bluemix to analyze a twitter data stream. Through that stream we will also plot graphs by using GraphX within the Spark service.

Prerequisites

You will need the following to complete this lab successfully,

- Browser for accessing IBM Bluemix and Twitter
- Active email ID for registering with Bluemix
- Registered and active accounts with Bluemix

Technology Involved

The following technologies will be covered in this lab,

- IBM Bluemix(PaaS)
- Twitter Apps
- Apache Spark(Service in Bluemix)
- Scala and Python Commands

Lab Steps

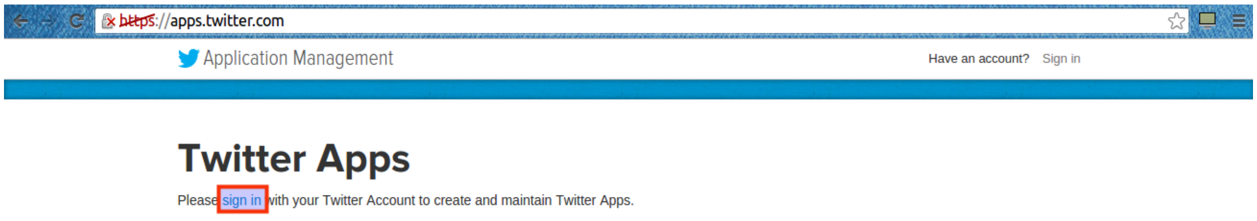
So, let us get started with the lab!

#1 | Create an application in Twitter

The first step will be to make sure that we have access to Twitter.

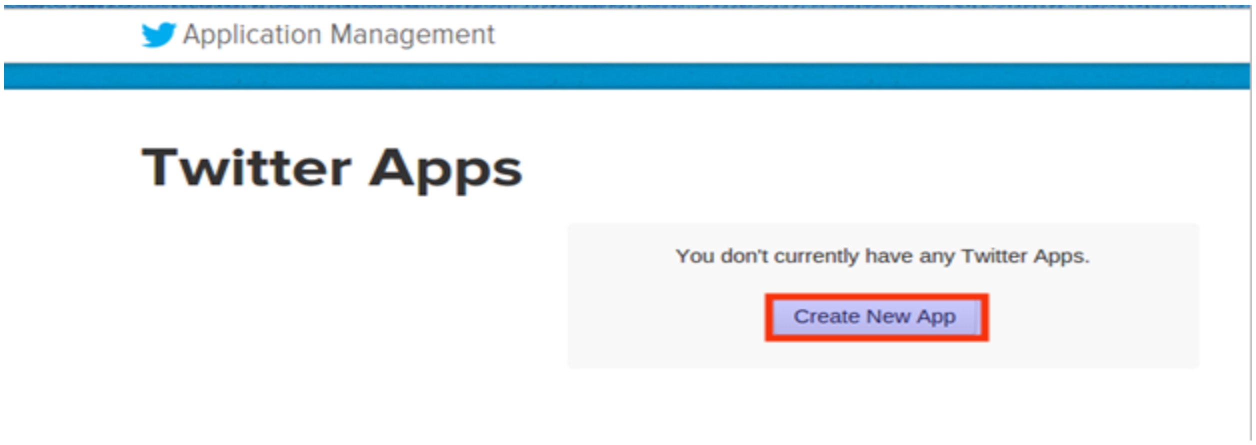
Login to Twitter Apps at https://apps.twitter.com/ (or) Register today at https://twitter.com/signup

Note : Twitter account should be registered with mobile in order to create applications

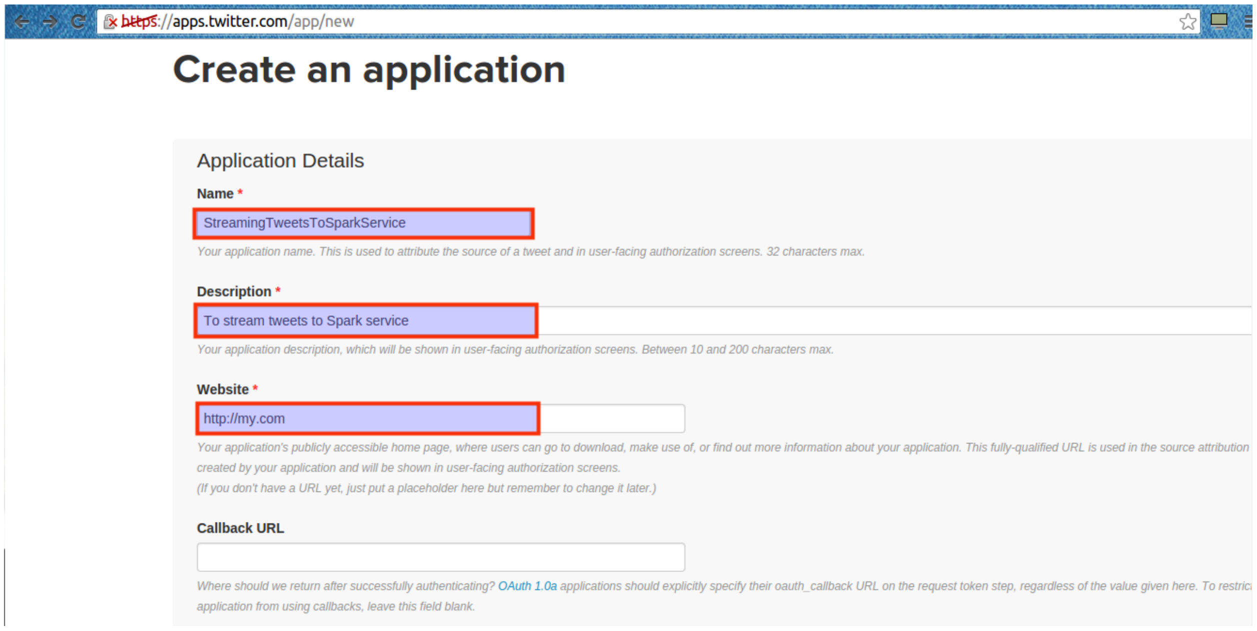


Please sign in with your Twitter Account to create and maintain Twitter Apps.

After you login, Click on **"Create New App"**.



You don't currently have any Twitter Apps.


~~https~~://apps.twitter.com/app/new

Create an application

Application Details

Name *

StreamingTweetsToSparkService

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

To stream tweets to Spark service

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

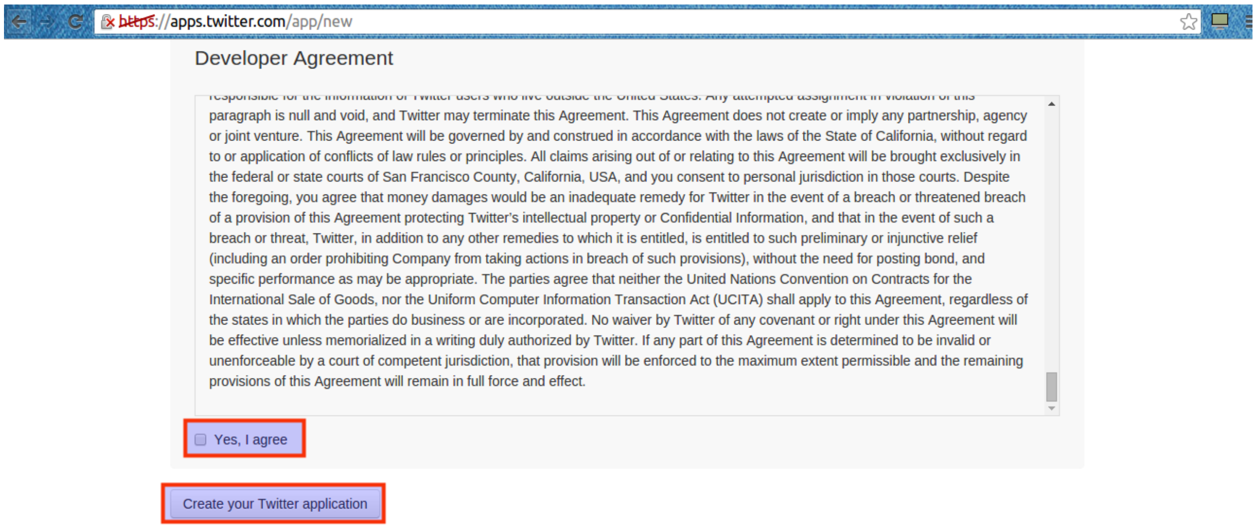
http://my.com

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

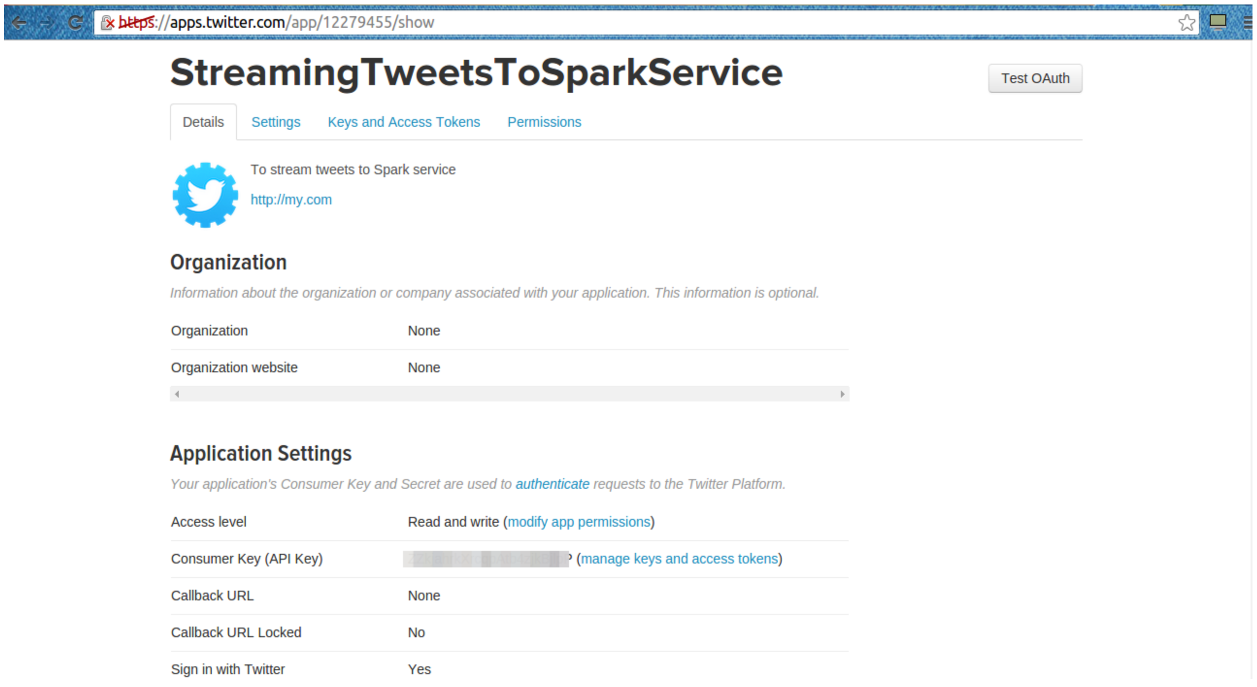
Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restric application from using callbacks, leave this field blank.

Scroll down and check "Yes, I agree" box and click on **"Create Your Twitter Application"**.


~~https~~://apps.twitter.com/app/new

Developer Agreement

paragraph is null and void, and Twitter may terminate this Agreement. This Agreement does not create or imply any partnership, agency or joint venture. This Agreement will be governed by and construed in accordance with the laws of the State of California, without regard to or application of conflicts of law rules or principles. All claims arising out of or relating to this Agreement will be brought exclusively in the federal or state courts of San Francisco County, California, USA, and you consent to personal jurisdiction in those courts. Despite the foregoing, you agree that money damages would be an inadequate remedy for Twitter in the event of a breach or threatened breach of a provision of this Agreement protecting Twitter's intellectual property or Confidential Information, and that in the event of such a breach or threat, Twitter, in addition to any other remedies to which it is entitled, is entitled to such preliminary or injunctive relief (including an order prohibiting Company from taking actions in breach of such provisions), without the need for posting bond, and specific performance as may be appropriate. The parties agree that neither the United Nations Convention on Contracts for the International Sale of Goods, nor the Uniform Computer Information Transaction Act (UCITA) shall apply to this Agreement, regardless of the states in which the parties do business or are incorporated. No waiver by Twitter of any covenant or right under this Agreement will be effective unless memorialized in a writing duly authorized by Twitter. If any part of this Agreement is determined to be invalid or unenforceable by a court of competent jurisdiction, that provision will be enforced to the maximum extent permissible and the remaining provisions of this Agreement will remain in full force and effect.


https://apps.twitter.com/app/12279455/show

To stream tweets to Spark service

http://my.com

Organization

Information about the organization or company associated with your application. This information is optional.

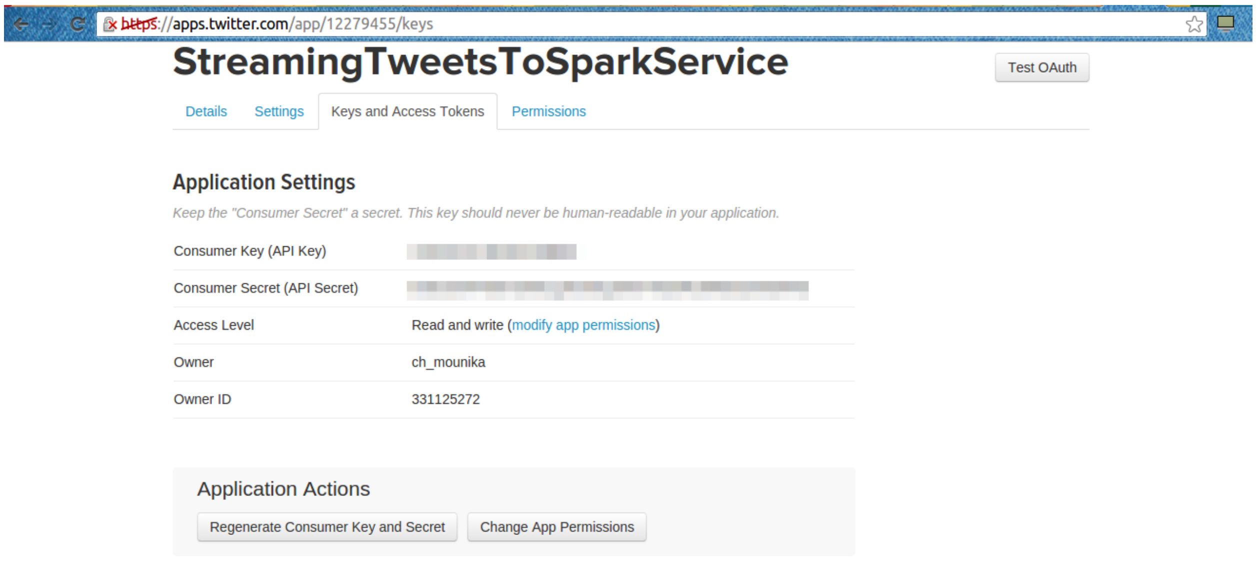
Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	[redacted] (manage keys and access tokens)
Callback URL	None
Callback URL Locked	No
Sign in with Twitter	Yes

Consumer Secret and Consumer Key will be generated. Go to **keys and access tokens** tab.


https://apps.twitter.com/app/12279455/keys

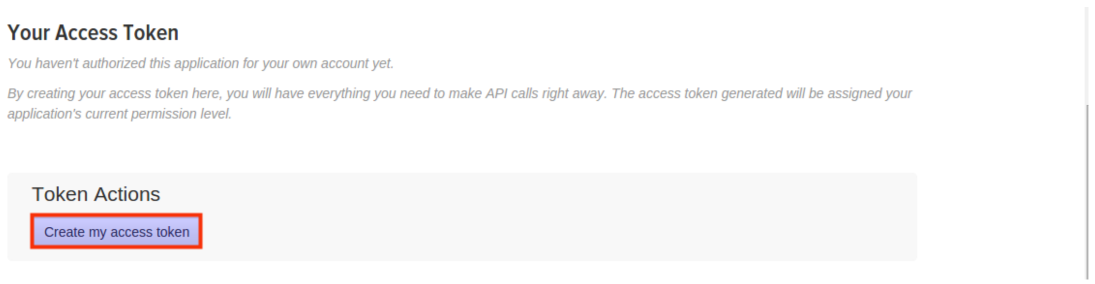
Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read and write (modify app permissions)
Owner	ch_mounika
Owner ID	331125272

Application Actions

Scroll down and click on **"Create my access token"**.



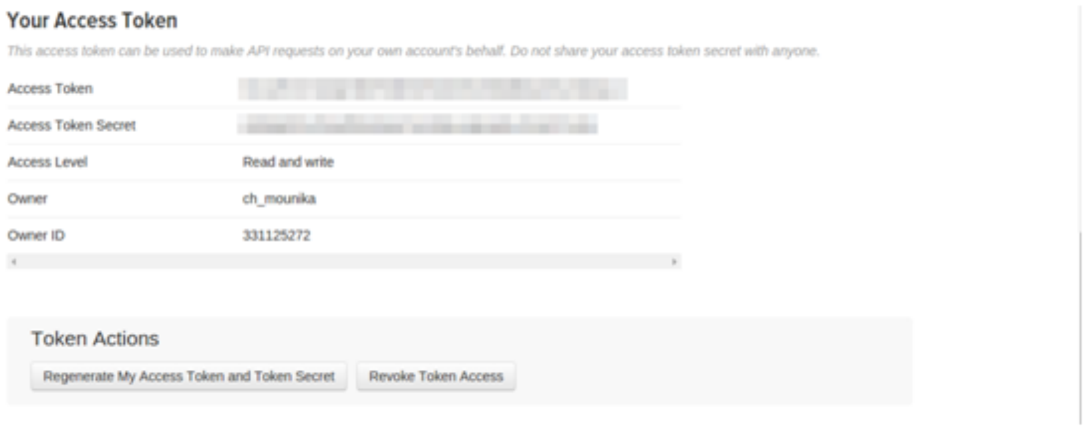
Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

Token keys will be generated as follows,



Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]
Access Level	Read and write
Owner	ch_mounika
Owner ID	331125272

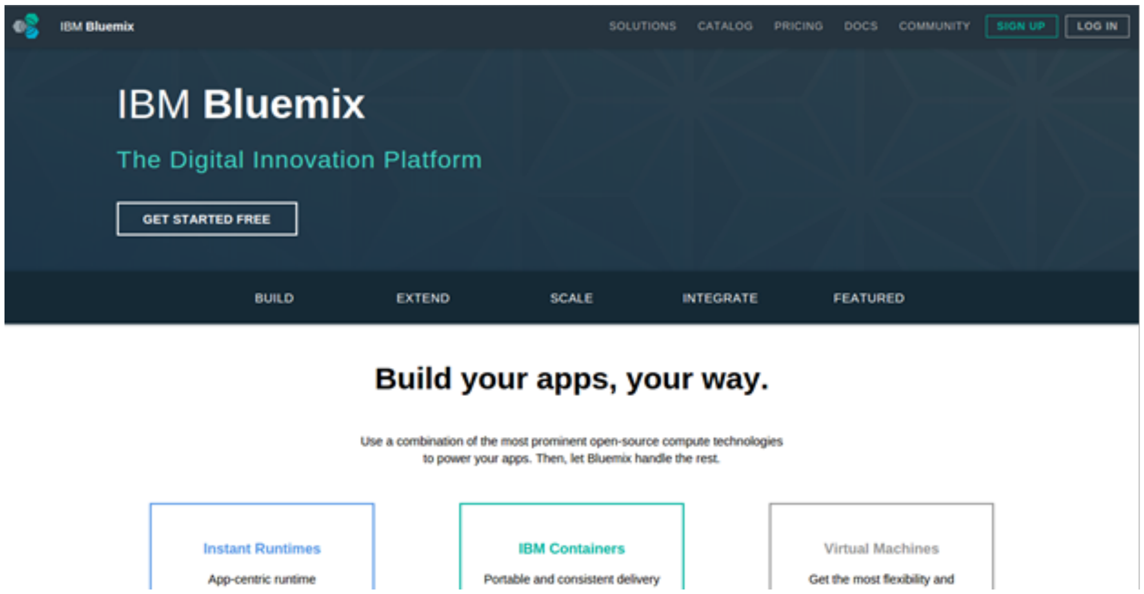
Token Actions

Please note down the Consumer keys and Access tokens for later purpose.

#2 | Access Bluemix

Make sure that we have access to the IBM Bluemix Console with either the free trial option (or) the paid subscription option.

Login to Bluemix at http://bluemix.net (or) Register today at https://console.ng.bluemix.net/registration/



IBM **Bluemix**

The Digital Innovation Platform

Build your apps, your way.

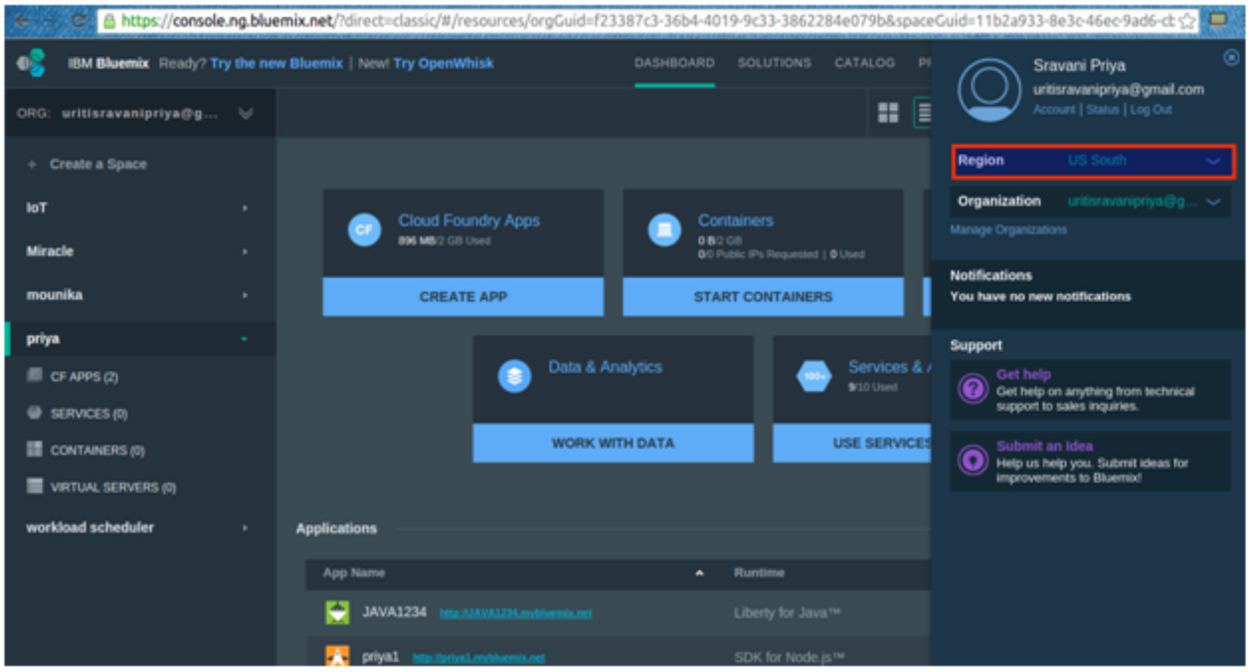
Use a combination of the most prominent open-source compute technologies to power your apps. Then, let Bluemix handle the rest.

App-centric runtime

Portable and consistent delivery

Get the most flexibility and

After you login, you can see the dashboard where you can take a look at your applications and services. You can also go to the profile icon at the top and change which Cloud Availability Region you are working in. Make sure you select **US South** region, as **Apache Spark Service** is only available in that region.



https://console.ng.bluemix.net/?direct=classic/#/resources/orgGuid=f23387c3-36b4-4019-9c33-3862284e079b&spaceGuid=11b2a933-8e3c-46ec-9ad6-c

ORG: uritisravanipriya@g...

896 MB/2 GB Used

0 B/2 GB
0/0 Public IPs Requested | 0 Used

9/10 Used

Sravani Priya
uritisravanipriya@gmail.com

Region: US South

Organization: uritisravanipriya@g...

You have no new notifications

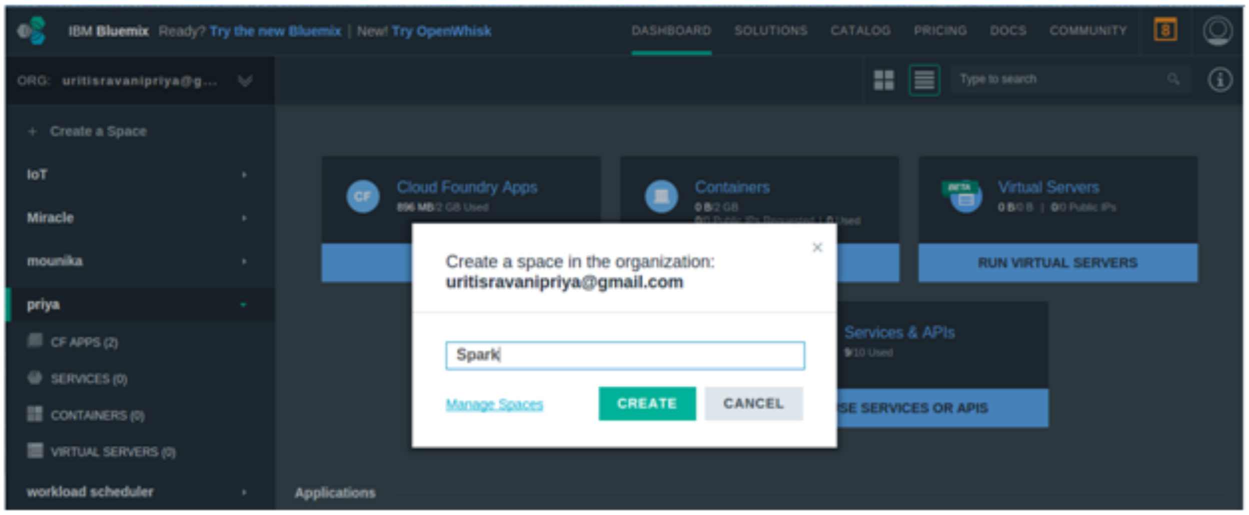
Get help on anything from technical support to sales inquiries.

Help us help you. Submit ideas for improvements to Bluemix!

App Name		Runtime
JAVA1234	http://JAVA1234.mybluemix.net	Liberty for Java™
priya1	http://priya1.mybluemix.net	SDK for Node.js™

#3 | Create Space

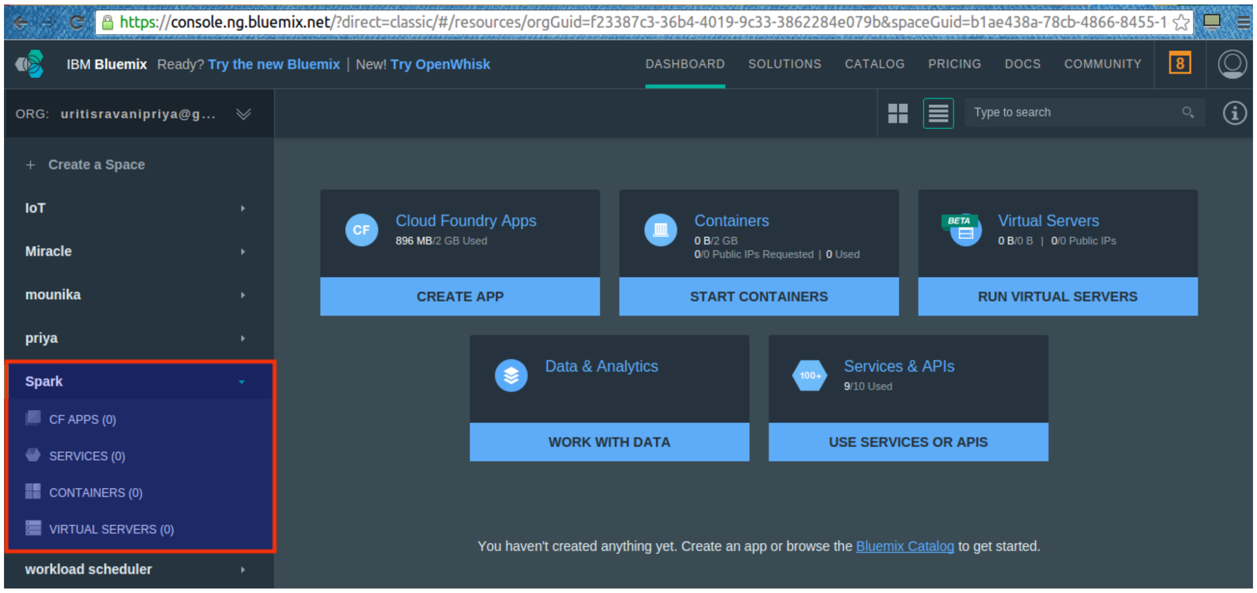
Click on **"Create a Space"** on left side of the Dashboard to create a space in Bluemix Cloud. Give any name and click on "create".



Create a space in the organization:
uritisravanipriya@gmail.com

Spark

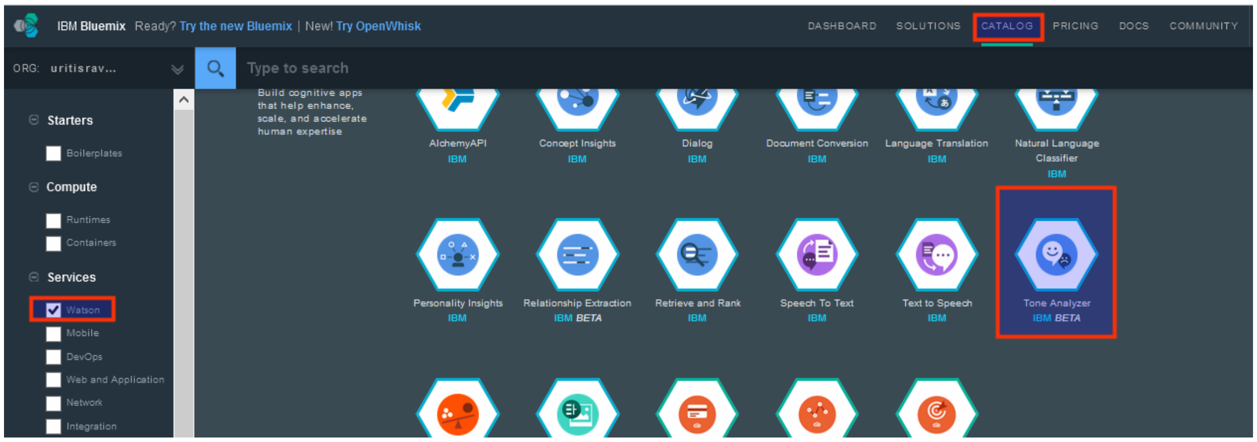
Once the space is created, you can find it at the left side menu of the Dashboard.



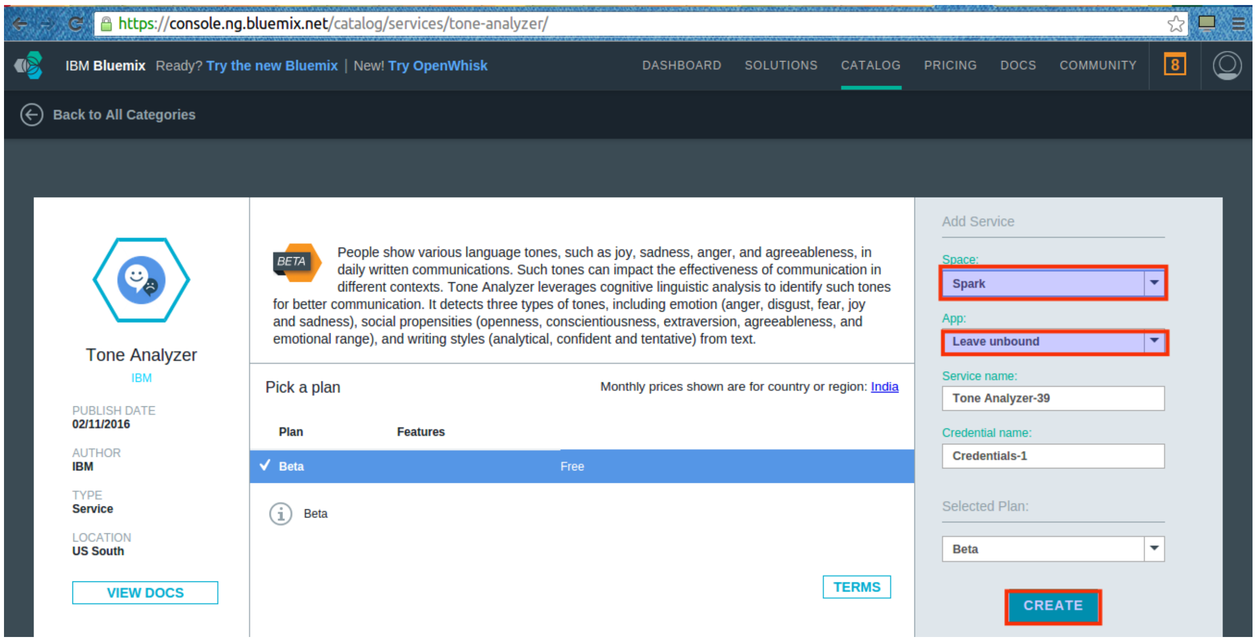
You haven't created anything yet. Create an app or browse the Bluemix Catalog to get started.

#4 | Create Watson Tone Analyzer Service

Go to “Catalog” and select “Watson” in the left side menu. Click on **“Tone Analyzer”** service.



Choose your created space in the “Space” field. Choose “Leave Unbound” in the “App” field and click on “Create”.


https://console.ng.bluemix.net/catalog/services/tone-analyzer/
IBM Bluemix Ready? Try the new Bluemix | New! Try OpenWhisk
DASHBOARD
SOLUTIONS
CATALOG
PRICING
DOCS
COMMUNITY
Back to All Categories
BETA
VIEW DOCS
TERMS
CREATE

Tone Analyzer
IBM

PUBLISH DATE
02/11/2016

AUTHOR
IBM

TYPE
Service

LOCATION
US South

People show various language tones, such as joy, sadness, anger, and agreeableness, in daily written communications. Such tones can impact the effectiveness of communication in different contexts. Tone Analyzer leverages cognitive linguistic analysis to identify such tones for better communication. It detects three types of tones, including emotion (anger, disgust, fear, joy and sadness), social propensities (openness, conscientiousness, extraversion, agreeableness, and emotional range), and writing styles (analytical, confident and tentative) from text.

Pick a plan

Monthly prices shown are for country or region: India

Plan	Features
✓ Beta	Free

Beta

Add Service

Space: Spark

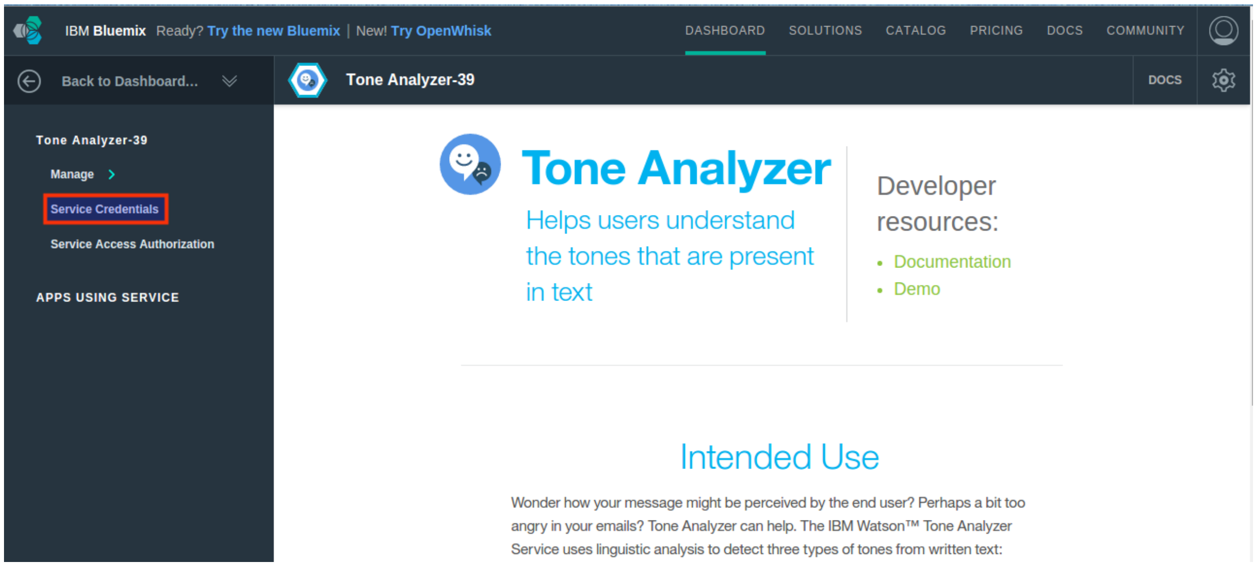
App: Leave unbound

Service name: Tone Analyzer-39

Credential name: Credentials-1

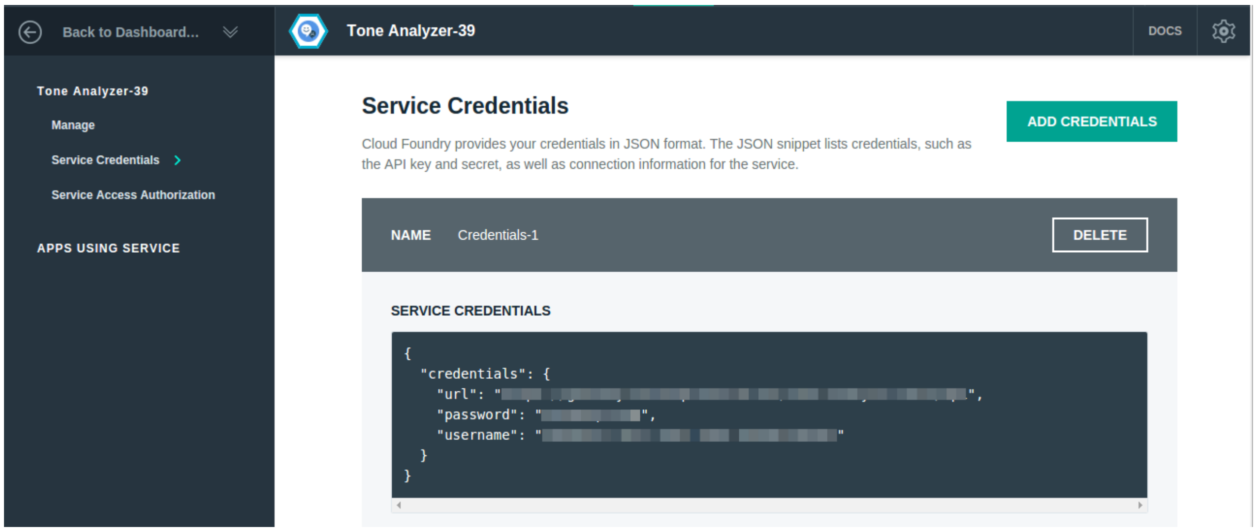
Selected Plan: Beta

Click on “Service Credentials”.



Wonder how your message might be perceived by the end user? Perhaps a bit too angry in your emails? Tone Analyzer can help. The IBM Watson™ Tone Analyzer Service uses linguistic analysis to detect three types of tones from written text:

Copy the credentials for later purpose.



Service Credentials

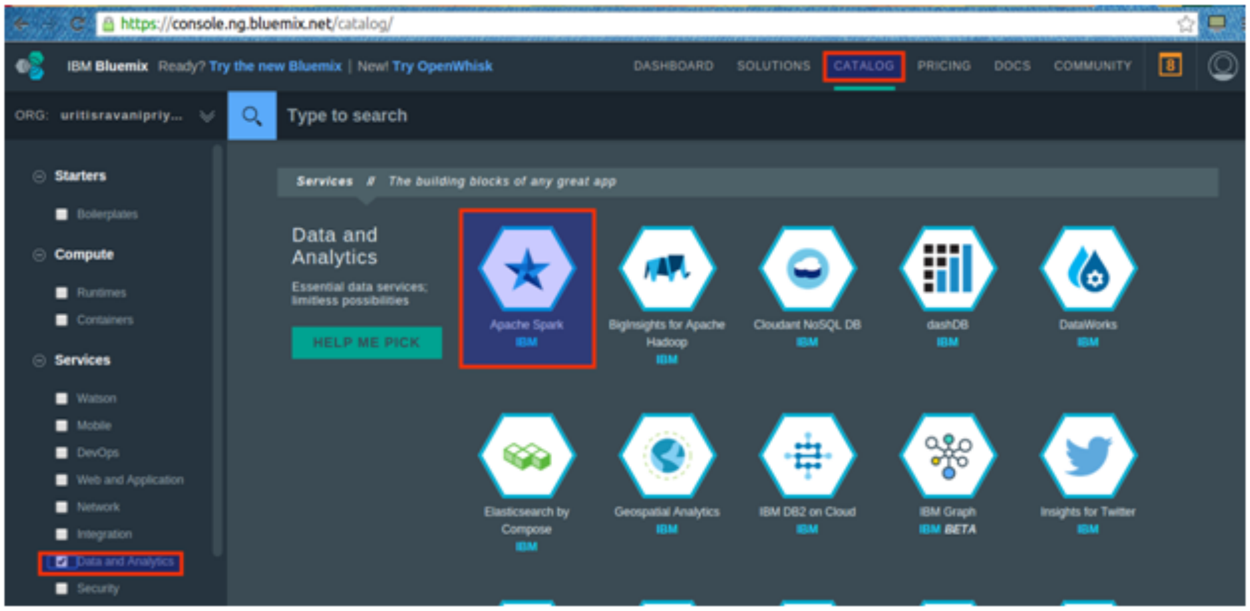
Cloud Foundry provides your credentials in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

SERVICE CREDENTIALS

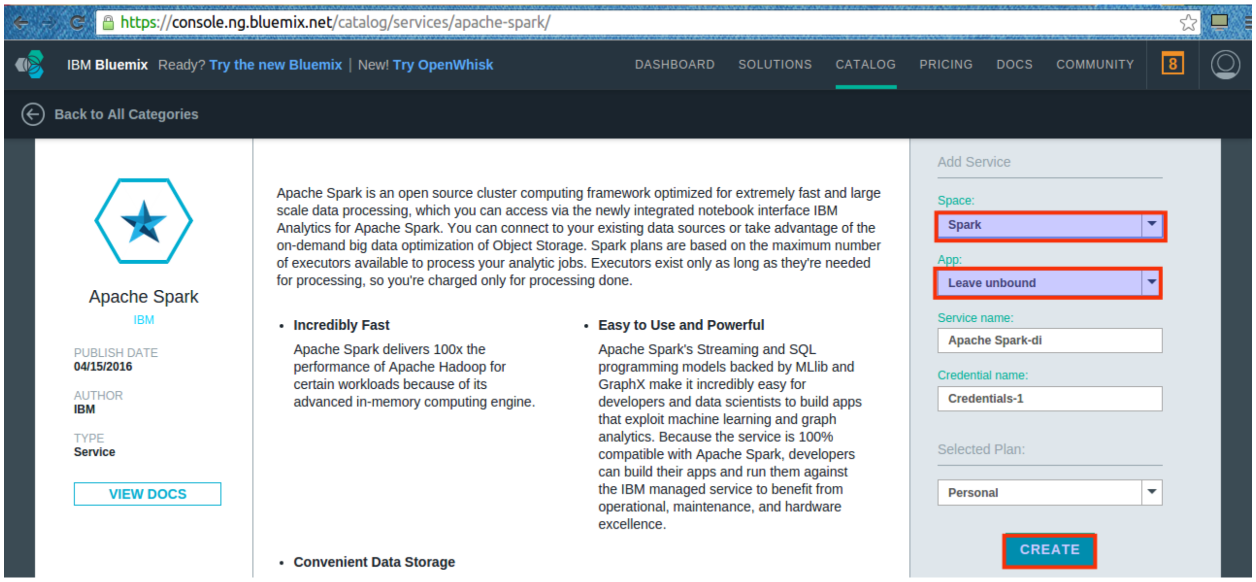
```
{
  "credentials": {
    "url": "[redacted]",
    "password": "[redacted]",
    "username": "[redacted]"
  }
}
```

#5 | Create Apache Spark Service

Go to “Catalog” and select “Data and Analytics” in the left side menu. Click on **“Apache Spark”** service.



Choose your created space in the “Space” field. Choose “Leave Unbound” in the “App” field and click on “Create”.


https://console.ng.bluemix.net/catalog/services/apache-spark/
IBM Bluemix Ready? Try the new Bluemix | New! Try OpenWhisk
DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY
Back to All Categories
Apache Spark
IBM
VIEW DOCS
CREATE

PUBLISH DATE
04/15/2016

AUTHOR
IBM

TYPE
Service

Apache Spark is an open source cluster computing framework optimized for extremely fast and large scale data processing, which you can access via the newly integrated notebook interface IBM Analytics for Apache Spark. You can connect to your existing data sources or take advantage of the on-demand big data optimization of Object Storage. Spark plans are based on the maximum number of executors available to process your analytic jobs. Executors exist only as long as they're needed for processing, so you're charged only for processing done.

- **Incredibly Fast**
  Apache Spark delivers 100x the performance of Apache Hadoop for certain workloads because of its advanced in-memory computing engine.
- **Easy to Use and Powerful**
  Apache Spark's Streaming and SQL programming models backed by MLlib and GraphX make it incredibly easy for developers and data scientists to build apps that exploit machine learning and graph analytics. Because the service is 100% compatible with Apache Spark, developers can build their apps and run them against the IBM managed service to benefit from operational, maintenance, and hardware excellence.
- **Convenient Data Storage**

Add Service

Space: Spark

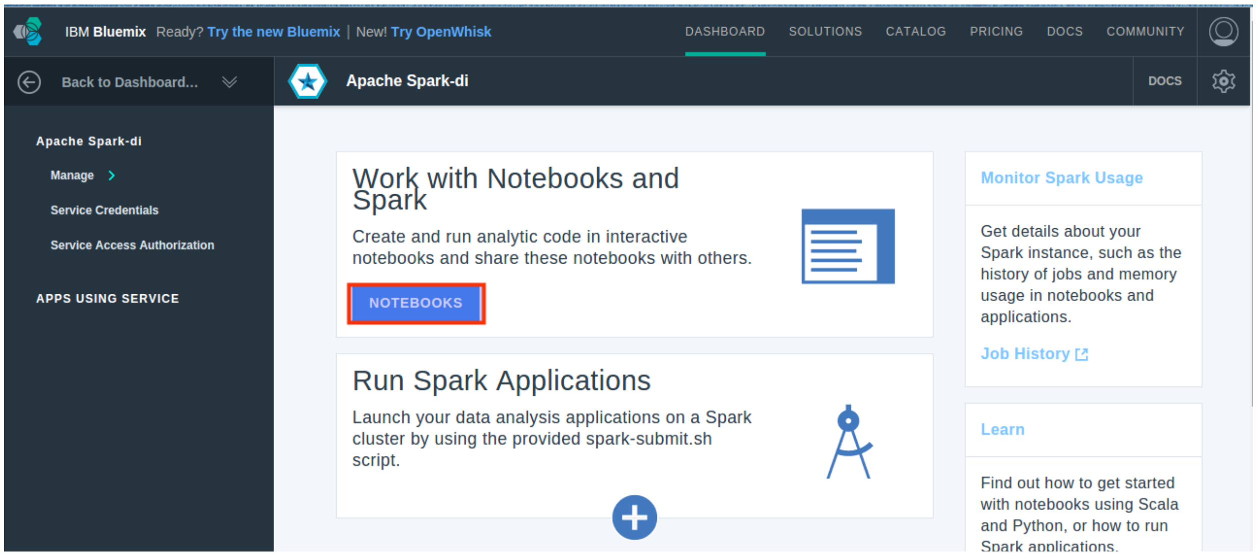
App: Leave unbound

Service name: Apache Spark-di

Credential name: Credentials-1

Selected Plan: Personal

Click on “Notebooks”.


IBM Bluemix Ready? Try the new Bluemix | New! Try OpenWhisk
DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY
Back to Dashboard...
Apache Spark-di
DOCS
Apache Spark-di
Manage
Service Credentials
Service Access Authorization
APPS USING SERVICE

Work with Notebooks and Spark

Create and run analytic code in interactive notebooks and share these notebooks with others.

NOTEBOOKS

Run Spark Applications

Launch your data analysis applications on a Spark cluster by using the provided spark-submit.sh script.

Monitor Spark Usage

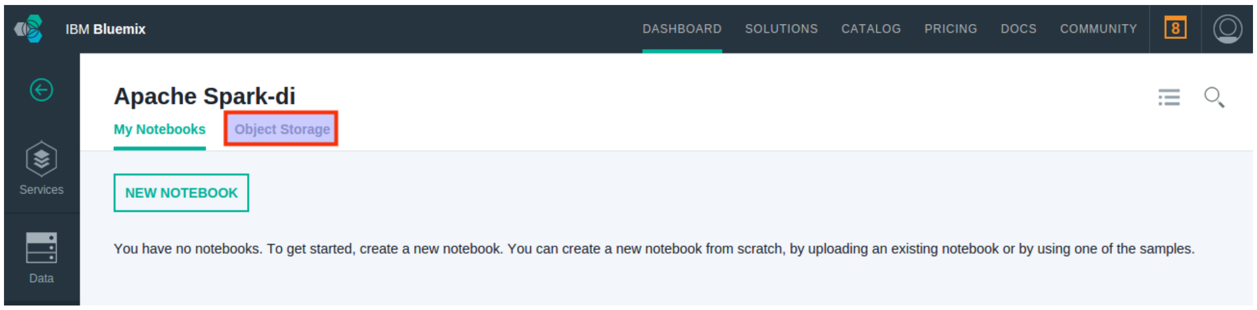
Get details about your Spark instance, such as the history of jobs and memory usage in notebooks and applications.

Job History

Learn

Find out how to get started with notebooks using Scala and Python, or how to run Spark applications.

Click on “Object Storage” tab.



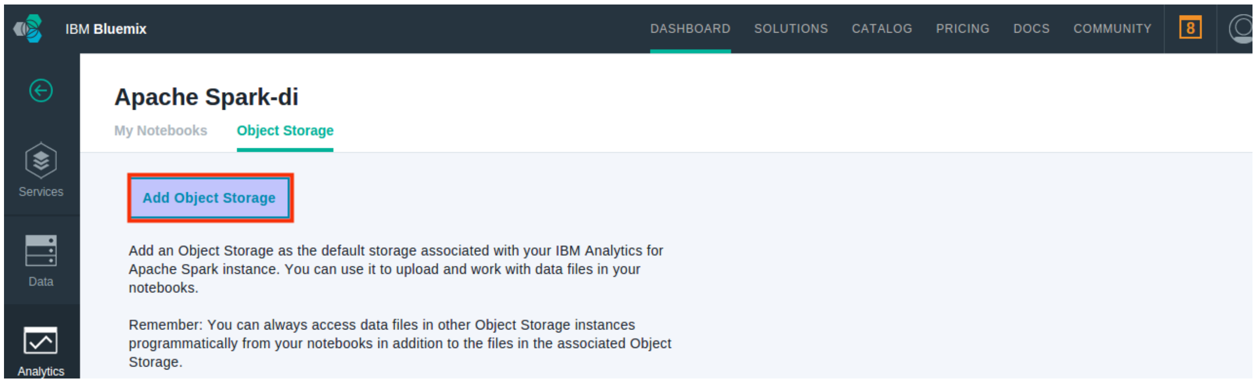
Apache Spark-di

My Notebooks Object Storage

NEW NOTEBOOK

You have no notebooks. To get started, create a new notebook. You can create a new notebook from scratch, by uploading an existing notebook or by using one of the samples.

Click on “Add Object Storage”.



Apache Spark-di

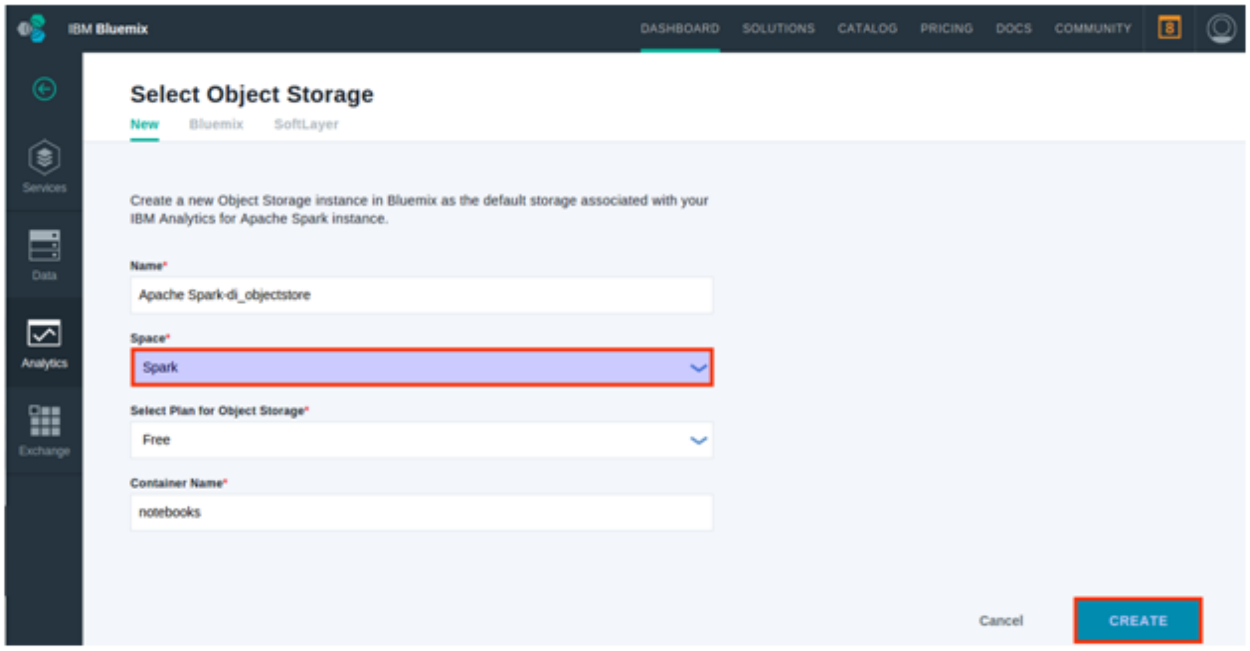
My Notebooks Object Storage

Add Object Storage

Add an Object Storage as the default storage associated with your IBM Analytics for Apache Spark instance. You can use it to upload and work with data files in your notebooks.

Remember: You can always access data files in other Object Storage instances programmatically from your notebooks in addition to the files in the associated Object Storage.

Select your space for “Space” field. Click on “Create”.



Select Object Storage

New Bluemix SoftLayer

Create a new Object Storage instance in Bluemix as the default storage associated with your IBM Analytics for Apache Spark instance.

Name*: Apache Spark-di_objectstore

Space*: Spark

Select Plan for Object Storage*: Free

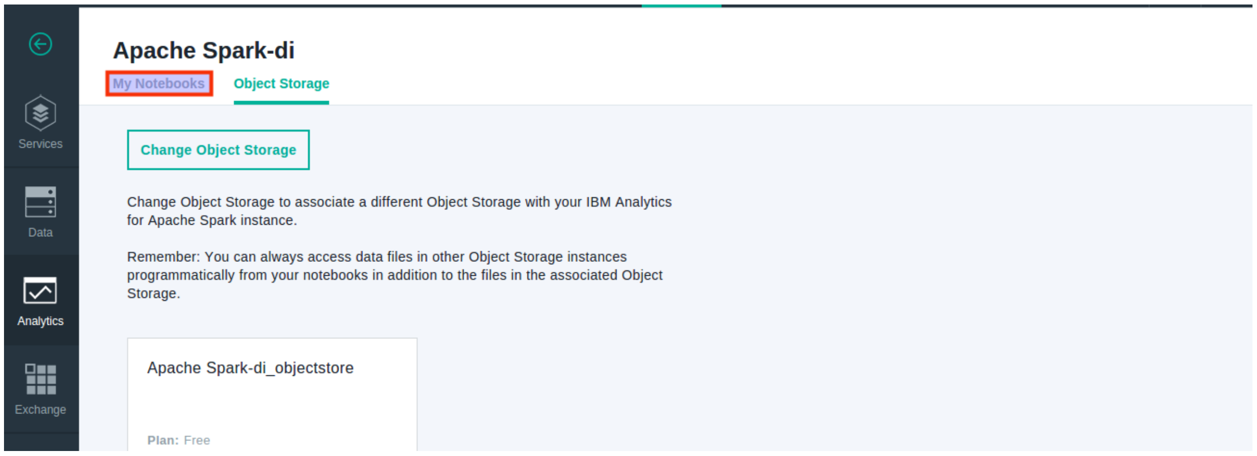
Container Name*: notebooks

Cancel CREATE

A new Object Storage will be created.

#6 | Create Scala Notebook

Click on **"My Notebooks"** to create a new notebook.



Apache Spark-di

My Notebooks Object Storage

Change Object Storage

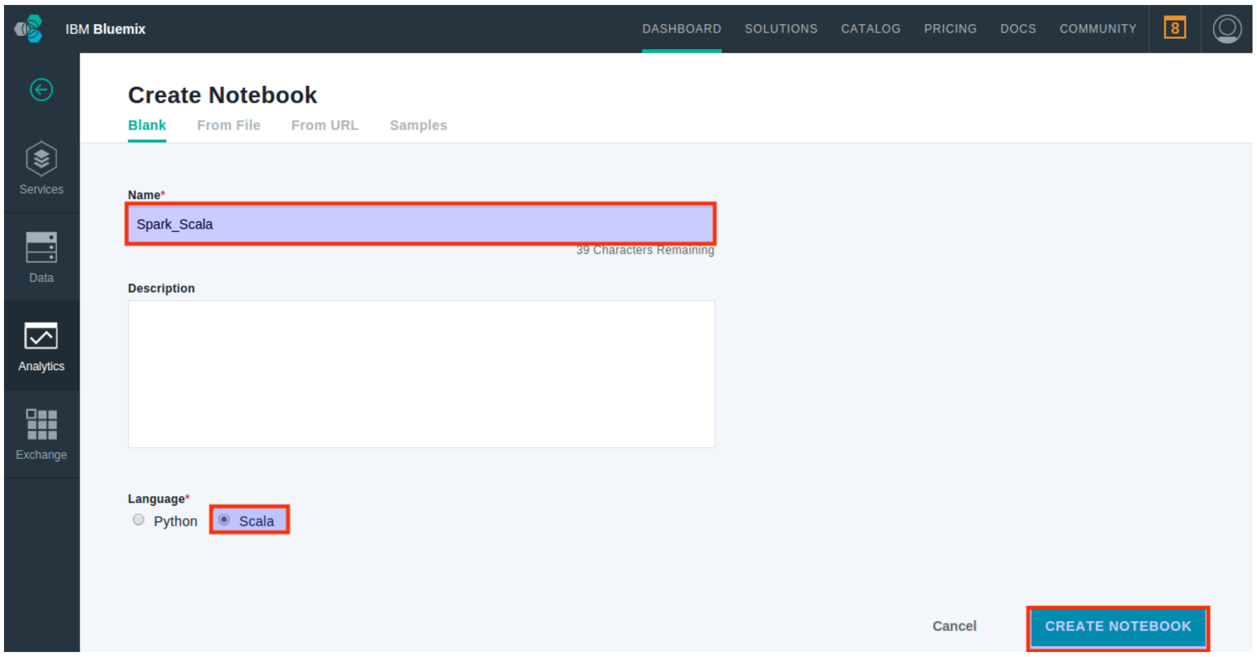
Change Object Storage to associate a different Object Storage with your IBM Analytics for Apache Spark instance.

Remember: You can always access data files in other Object Storage instances programmatically from your notebooks in addition to the files in the associated Object Storage.

Apache Spark-di_objectstore

Plan: Free

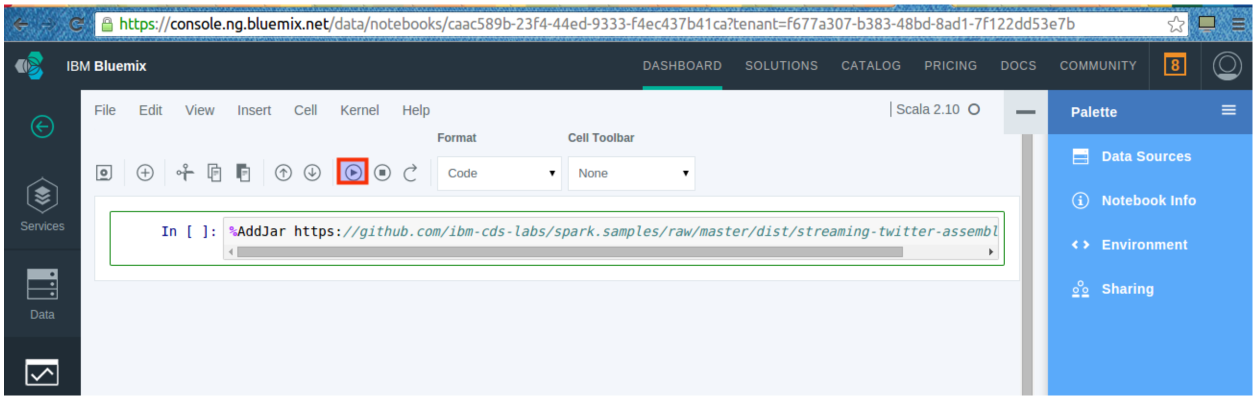
Give any name for "Name" field. Select "Scala" radio button for "Language" field. Click on "Create Notebook".



Name*: Spark_Scala
39 Characters Remaining
Description
Language*: Python [✓ Scala]

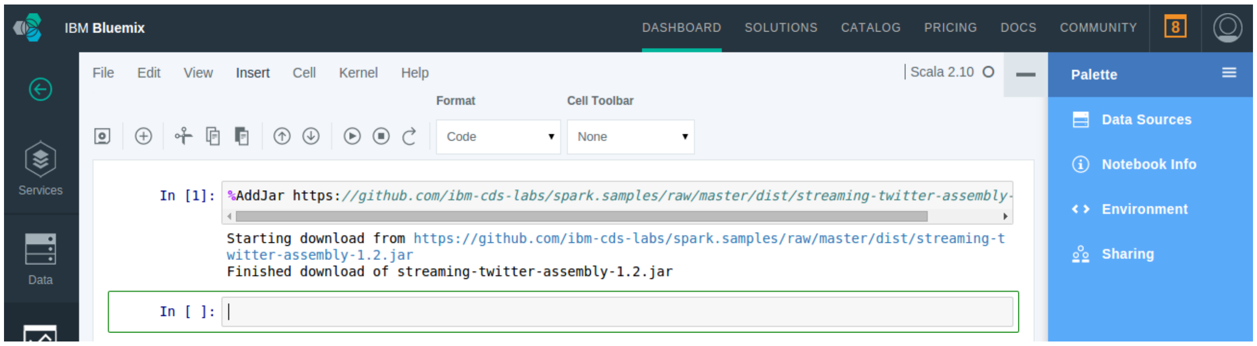
A Scala shell will be opened. In the first cell, enter the following command to install the application jar.

```
%AddJar https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-twitter-assembly-1.2.jar -f
```



```
In [ ]: %AddJar https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-twitter-assembl
```

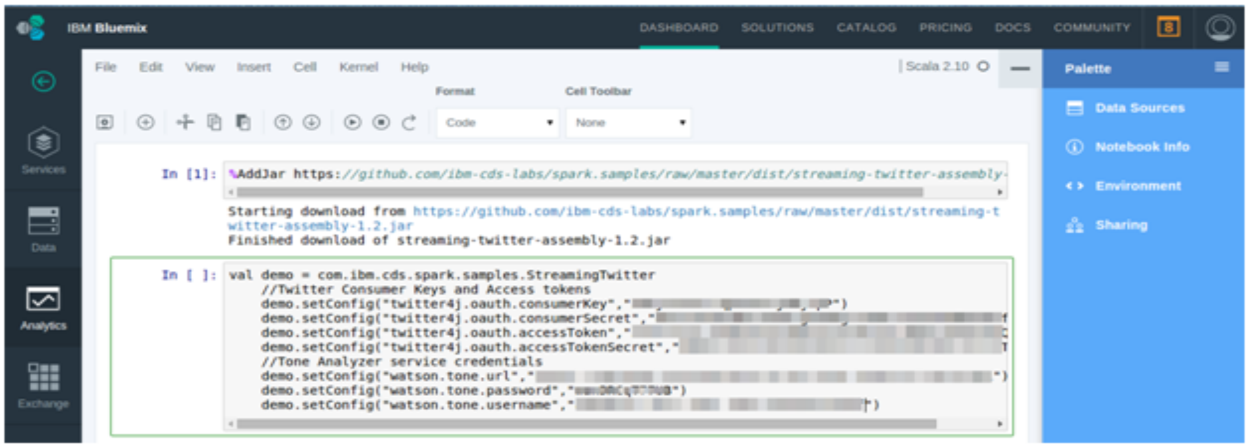
Click on run button to execute the cell.



```
In [1]: %AddJar https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-twitter-assembly-

Starting download from https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-t
witter-assembly-1.2.jar
Finished download of streaming-twitter-assembly-1.2.jar

In [ ]:
```

Once the jar is downloaded, configure the credentials needed to connect to Twitter and Watson Tone Analyzer service.

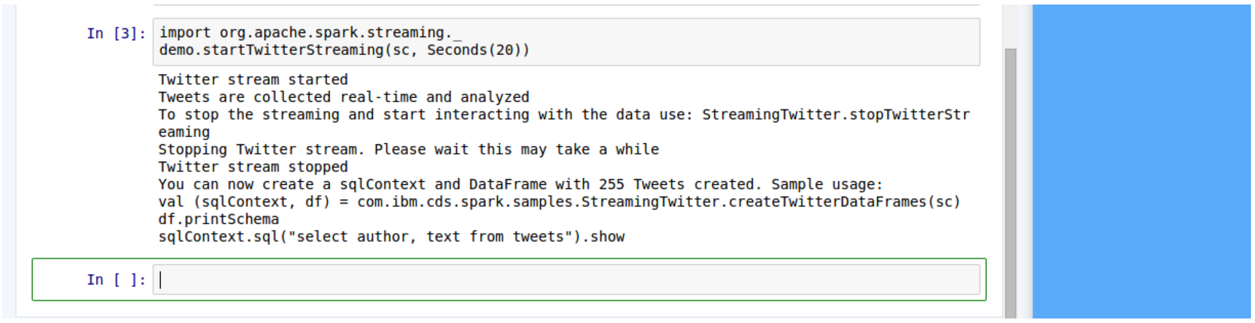
```
val demo = com.ibm.cds.spark.samples.StreamingTwitter

//Twitter Consumer Keys and Access tokens
demo.setConfig("twitter4j.oauth.consumerKey","<Consumer-Key>")
demo.setConfig("twitter4j.oauth.consumerSecret","<Consumer-
Secret>")
demo.setConfig("twitter4j.oauth.accessToken","<access-Token>")
demo.setConfig("twitter4j.oauth.accessTokenSecret","<access-
Token-Secret>")

//Tone Analyzer service credentials
demo.setConfig("watson.tone.url","<URL>")
demo.setConfig("watson.tone.password","<password>")
demo.setConfig("watson.tone.username","<user-name>")
```



```
In [1]: %AddJar https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-twitter-assembly-

Starting download from https://github.com/ibm-cds-labs/spark.samples/raw/master/dist/streaming-t
witter-assembly-1.2.jar
Finished download of streaming-twitter-assembly-1.2.jar

In [ ]: val demo = com.ibm.cds.spark.samples.StreamingTwitter
            //Twitter Consumer Keys and Access tokens
            demo.setConfig("twitter4j.oauth.consumerKey","[redacted]")
            demo.setConfig("twitter4j.oauth.consumerSecret","[redacted]
            demo.setConfig("twitter4j.oauth.accessToken","[redacted]
            demo.setConfig("twitter4j.oauth.accessTokenSecret","[redacted]
            //Tone Analyzer service credentials
            demo.setConfig("watson.tone.url","[redacted]")
            demo.setConfig("watson.tone.password","[redacted]")
            demo.setConfig("watson.tone.username","[redacted]")
```

To start streaming tweets from Twitter, run the following command,

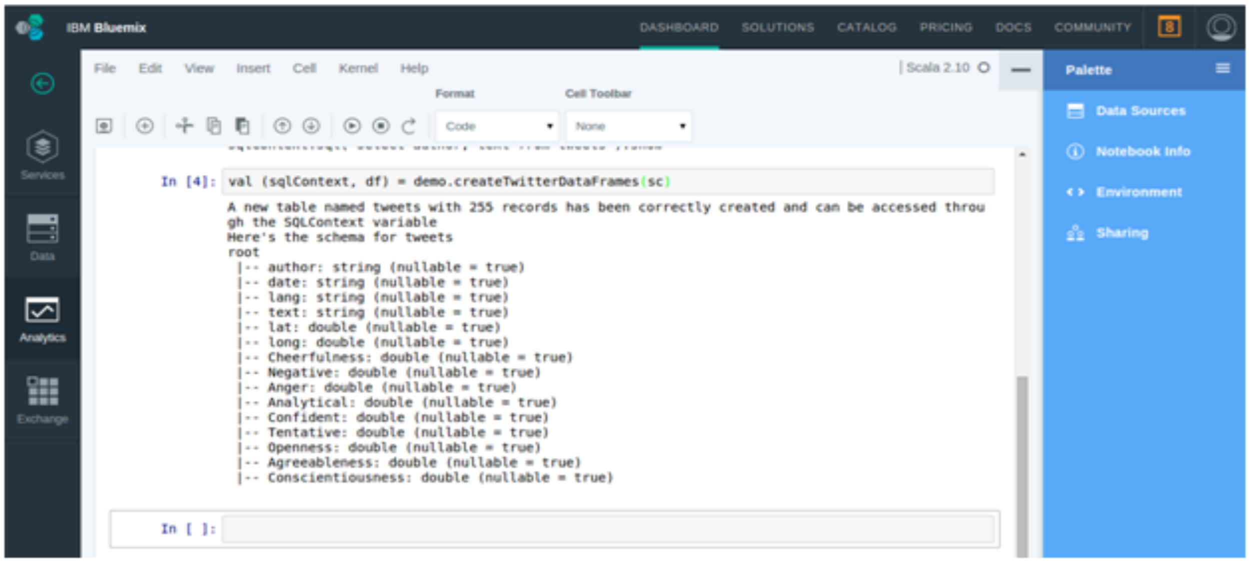
```
import org.apache.spark.streaming._
demo.startTwitterStreaming(sc, Seconds(20))
```



```
In [3]: import org.apache.spark.streaming._
        demo.startTwitterStreaming(sc, Seconds(20))

        Twitter stream started
        Tweets are collected real-time and analyzed
        To stop the streaming and start interacting with the data use: StreamingTwitter.stopTwitterStr
        eaming
        Stopping Twitter stream. Please wait this may take a while
        Twitter stream stopped
        You can now create a sqlContext and DataFrame with 255 Tweets created. Sample usage:
        val (sqlContext, df) = com.ibm.cds.spark.samples.StreamingTwitter.createTwitterDataFrames(sc)
        df.printSchema
        sqlContext.sql("select author, text from tweets").show

In [ ]: |
```

Once the stream stops, create a DataFrame using Spark SQL.

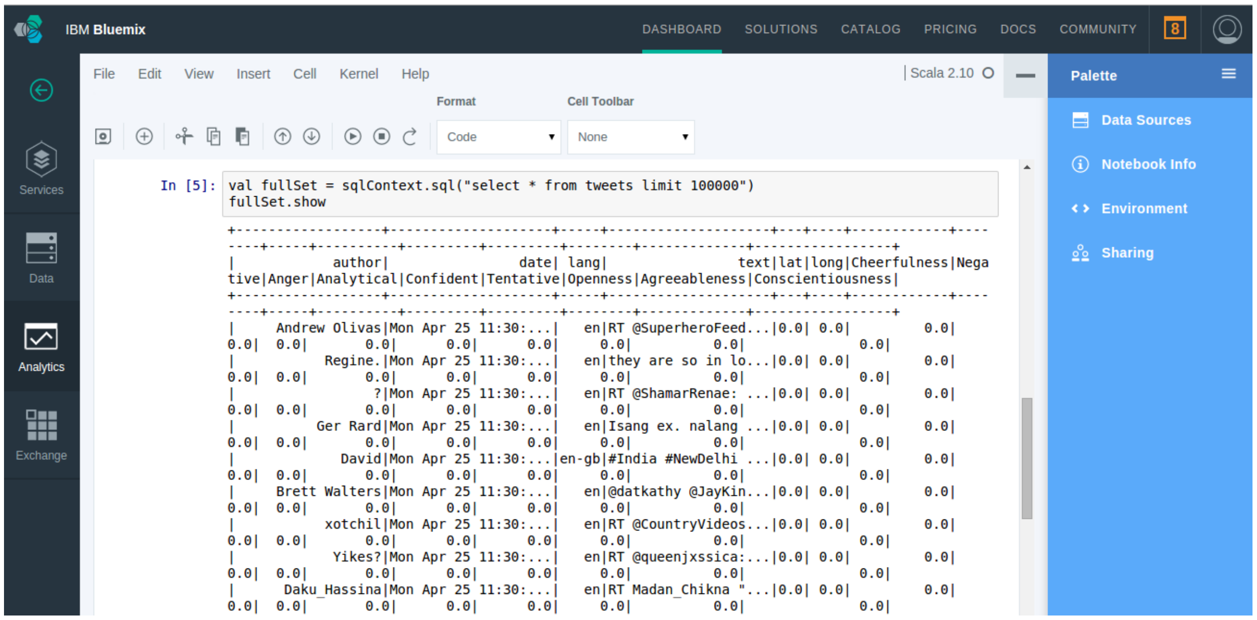
```
val(sqlContext,df) = demo.createTwitterDataFrames(sc)
```



```
In [4]: val (sqlContext, df) = demo.createTwitterDataFrames(sc)
        A new table named tweets with 255 records has been correctly created and can be accessed throu
        gh the SQLContext variable
        Here's the schema for tweets
        root
         |-- author: string (nullable = true)
         |-- date: string (nullable = true)
         |-- lang: string (nullable = true)
         |-- text: string (nullable = true)
         |-- lat: double (nullable = true)
         |-- long: double (nullable = true)
         |-- Cheerfulness: double (nullable = true)
         |-- Negative: double (nullable = true)
         |-- Anger: double (nullable = true)
         |-- Analytical: double (nullable = true)
         |-- Confident: double (nullable = true)
         |-- Tentative: double (nullable = true)
         |-- Openness: double (nullable = true)
         |-- Agreeableness: double (nullable = true)
         |-- Conscientiousness: double (nullable = true)

In [ ]:
```

Run the following command to display a sample of data.

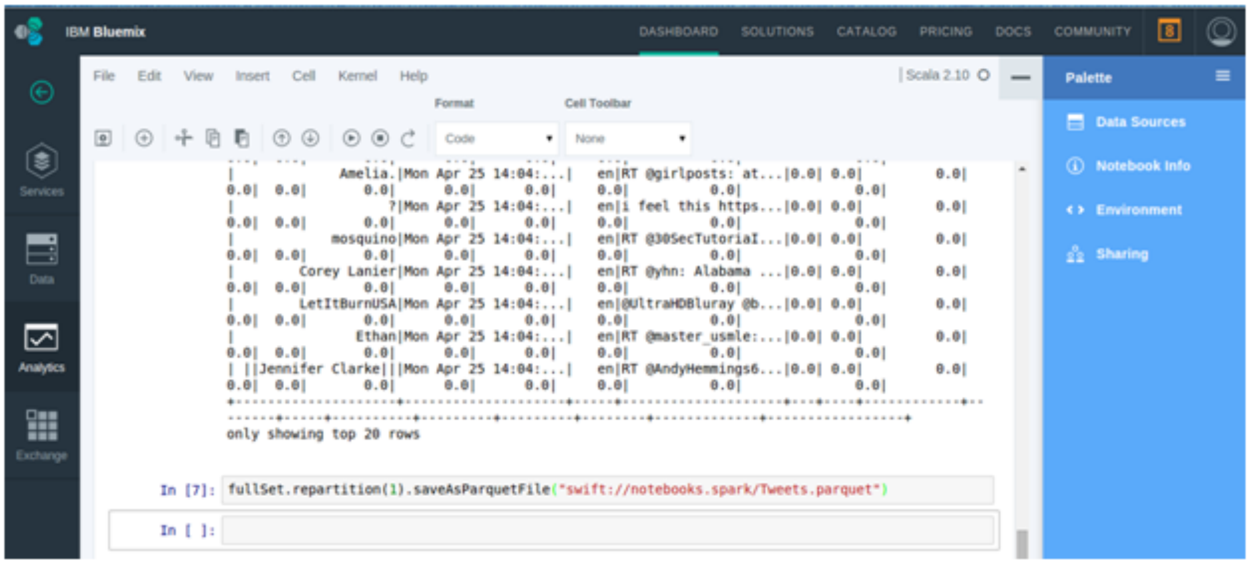
```
val fullSet = sqlContext.sql("select * from tweets limit
100000")
fullSet.show
```



```
In [5]: val fullSet = sqlContext.sql("select * from tweets limit 100000")
        fullSet.show

+-----------------+--------------------+-----+--------------------+---+----+------------+----
----+-----+----------+---------+---------+--------+--------------+-----------------+
|           author|                date| lang|                text|lat|long|Cheerfulness|Nega
tive|Anger|Analytical|Confident|Tentative|Openness|Agreeableness|Conscientiousness|
+-----------------+--------------------+-----+--------------------+---+----+------------+----
----+-----+----------+---------+---------+--------+--------------+-----------------+
|    Andrew Olivas|Mon Apr 25 11:30:...|   en|RT @SuperheroFeed...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|          Regine.|Mon Apr 25 11:30:...|   en|they are so in lo...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|                ?|Mon Apr 25 11:30:...|   en|RT @ShamarRenae: ...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|         Ger Rard|Mon Apr 25 11:30:...|   en|Isang ex. nalang ...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|            David|Mon Apr 25 11:30:...|en-gb|#India #NewDelhi ...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|    Brett Walters|Mon Apr 25 11:30:...|   en|@datkathy @JayKin...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|          xotchil|Mon Apr 25 11:30:...|   en|RT @CountryVideos...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|           Yikes?|Mon Apr 25 11:30:...|   en|RT @queenjxssica:...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|     Daku_Hassina|Mon Apr 25 11:30:...|   en|RT Madan_Chikna "...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
```

To save the dataset into a parquet file on Object Storage,

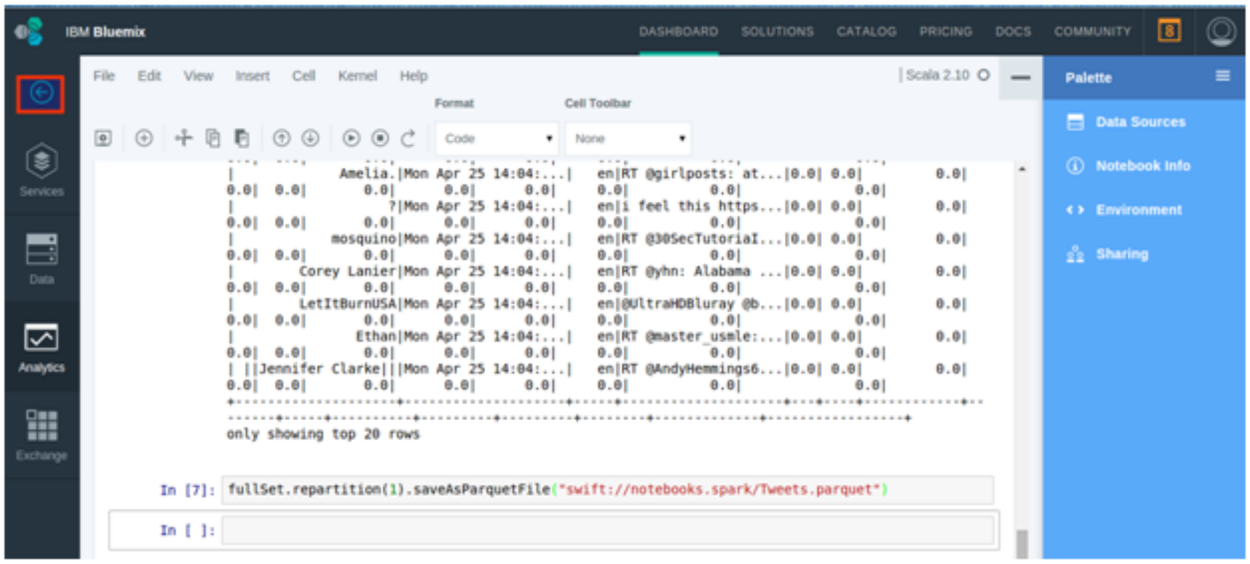
```
fullSet.repartition(1).saveAsParquetFile("swift://notebooks.spar
k/Tweets.parquet")
```



```
|          Amelia.|Mon Apr 25 14:04:...|   en|RT @girlposts: at...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|                ?|Mon Apr 25 14:04:...|   en|i feel this https...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|         mosquino|Mon Apr 25 14:04:...|   en|RT @30SecTutorial...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|     Corey Lanier|Mon Apr 25 14:04:...|   en|RT @yhn: Alabama ...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|     LetItBurnUSA|Mon Apr 25 14:04:...|   en|@UltraHDBluray @b...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
|            Ethan|Mon Apr 25 14:04:...|   en|RT @master_usmle:...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
| ||Jennifer Clarke|||Mon Apr 25 14:04:...|   en|RT @AndyHemmings6...|0.0| 0.0|         0.0|
0.0|  0.0|       0.0|      0.0|      0.0|     0.0|          0.0|              0.0|
+-----------------+--------------------+-----+--------------------+---+----+------------+----
----+-----+----------+---------+---------+--------+--------------+-----------------+
only showing top 20 rows

In [7]: fullSet.repartition(1).saveAsParquetFile("swift://notebooks.spark/Tweets.parquet")

In [ ]:
```

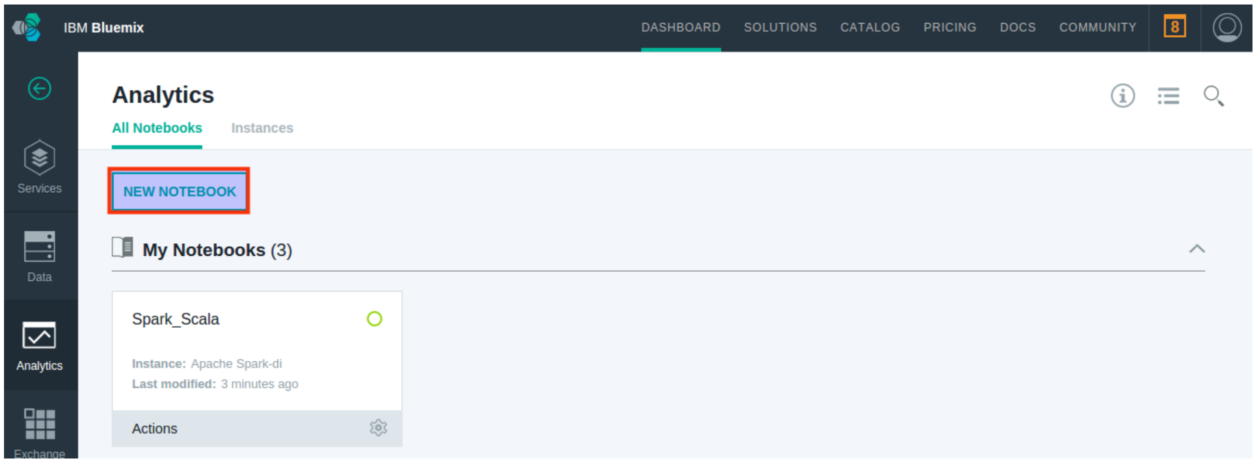
#7 | Analyzing data using Python Notebook

Go to Notebook page by clicking on back button on the upper left corner.

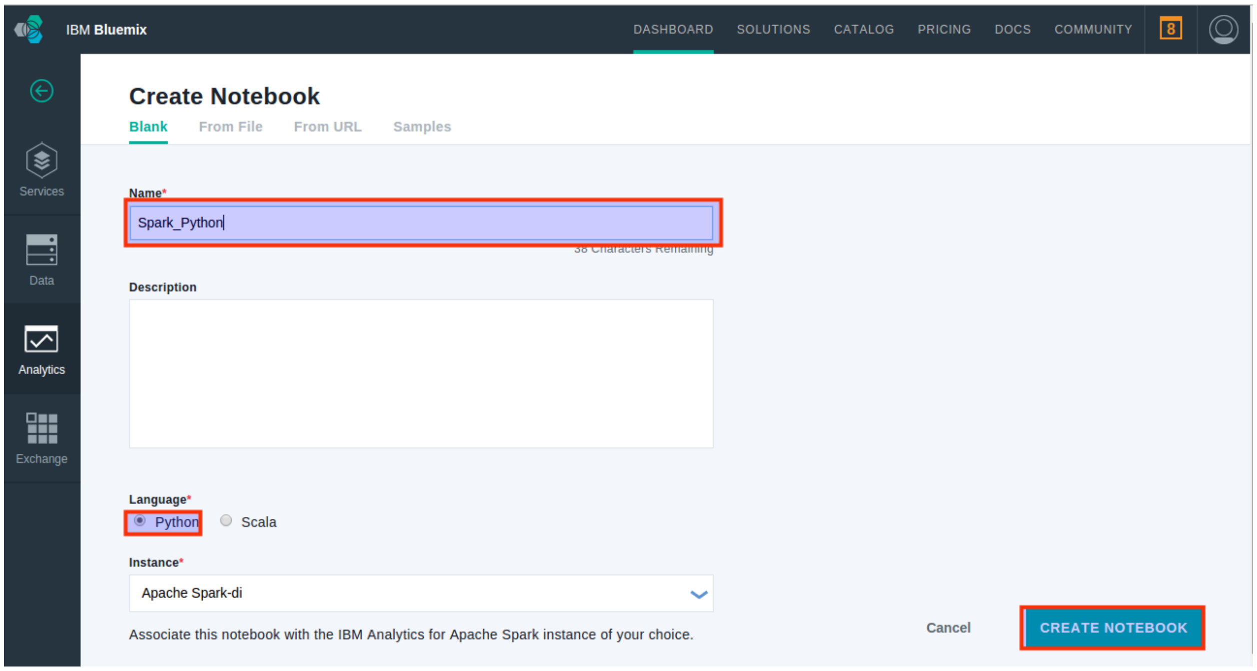


```
|              Amelia.|Mon Apr 25 14:04:...|   en|RT @girlposts: at...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
|                   ?|Mon Apr 25 14:04:...|   en|i feel this https...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
|            mosquino|Mon Apr 25 14:04:...|   en|RT @30SecTutorial...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
|        Corey Lanier|Mon Apr 25 14:04:...|   en|RT @yhn: Alabama ...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
|        LetItBurnUSA|Mon Apr 25 14:04:...|   en|@UltraHDBluray @b...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
|               Ethan|Mon Apr 25 14:04:...|   en|RT @master_usmle:...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
| ||Jennifer Clarke|||Mon Apr 25 14:04:...|   en|RT @AndyHemmings6...|0.0| 0.0|          0.0|
0.0|  0.0|        0.0|       0.0|       0.0|     0.0|          0.0|            0.0|
+--------------------+--------------------+-----+--------------------+---+----+--------------+--
-----+-----+----------+----------+----------+--------+-------------+------------------+
only showing top 20 rows

In [7]: fullSet.repartition(1).saveAsParquetFile("swift://notebooks.spark/Tweets.parquet")

In [ ]:
```

Click on **"New Notebook".**



Give any name for **"Name"** field. Select the **"Python"** radio button and click on **"Create Notebook"**.



Create Notebook

Blank | From File | From URL | Samples

Name*: Spark_Python

38 Characters Remaining

Description

Language*: [✓ Python] Scala

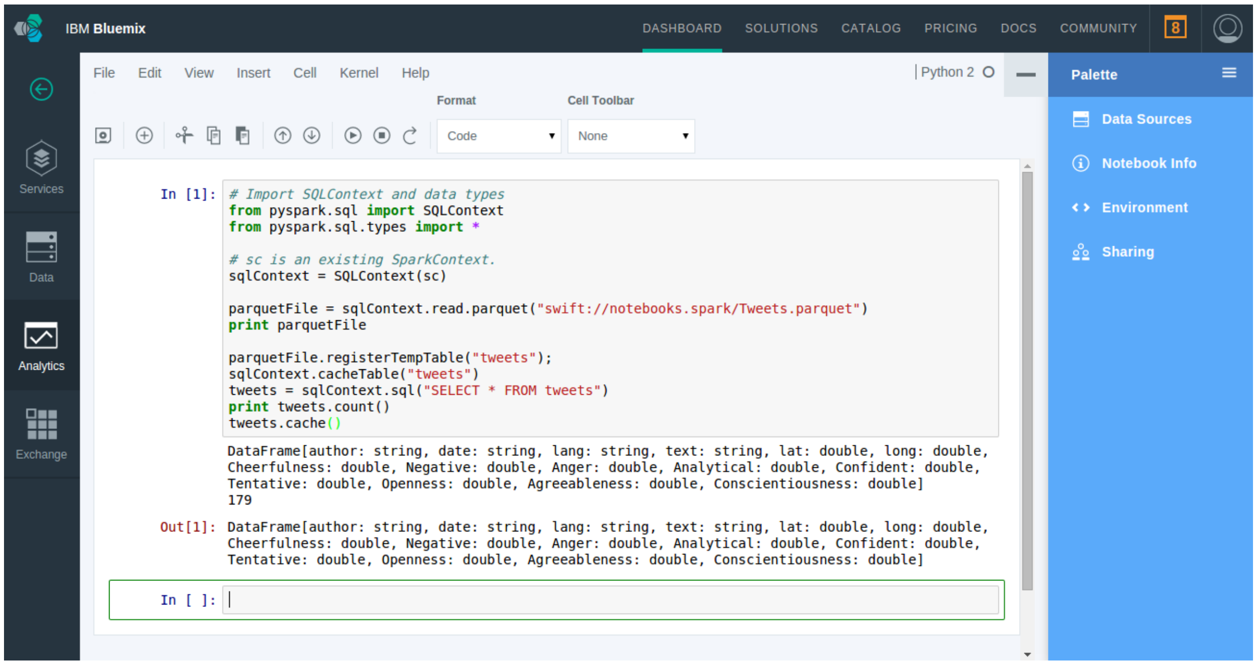
Instance*: Apache Spark-di

Associate this notebook with the IBM Analytics for Apache Spark instance of your choice.

Cancel | CREATE NOTEBOOK

A Python shell will appear. Load the data from Object Storage and create a DataFrame with the entire dataset by using the following command,

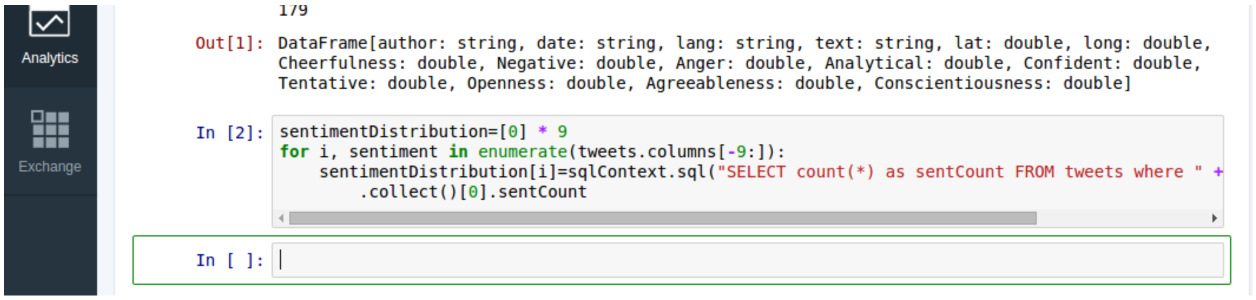
```
# Import SQLContext and data types
from pyspark.sql import SQLContext
from pyspark.sql.types import *
# sc is an existing SparkContext.
sqlContext = SQLContext(sc)
parquetFile =
sqlContext.read.parquet("swift://notebooks.spark/Tweets.parquet"
)
print parquetFile
parquetFile.registerTempTable("tweets");
sqlContext.cacheTable("tweets")
tweets = sqlContext.sql("SELECT * FROM tweets")
print tweets.count()
tweets.cache()
```



```
In [1]: # Import SQLContext and data types
        from pyspark.sql import SQLContext
        from pyspark.sql.types import *

        # sc is an existing SparkContext.
        sqlContext = SQLContext(sc)

        parquetFile = sqlContext.read.parquet("swift://notebooks.spark/Tweets.parquet")
        print parquetFile

        parquetFile.registerTempTable("tweets");
        sqlContext.cacheTable("tweets")
        tweets = sqlContext.sql("SELECT * FROM tweets")
        print tweets.count()
        tweets.cache()
        DataFrame[author: string, date: string, lang: string, text: string, lat: double, long: double,
        Cheerfulness: double, Negative: double, Anger: double, Analytical: double, Confident: double,
        Tentative: double, Openness: double, Agreeableness: double, Conscientiousness: double]
        179

Out[1]: DataFrame[author: string, date: string, lang: string, text: string, lat: double, long: double,
        Cheerfulness: double, Negative: double, Anger: double, Analytical: double, Confident: double,
        Tentative: double, Openness: double, Agreeableness: double, Conscientiousness: double]

In [ ]:
```

Now we can start analyzing the data by create an array that will hold the count for each sentiment. For each sentiment, run a SQL query that counts the number of tweets for which the sentiment score is greater than 60%.

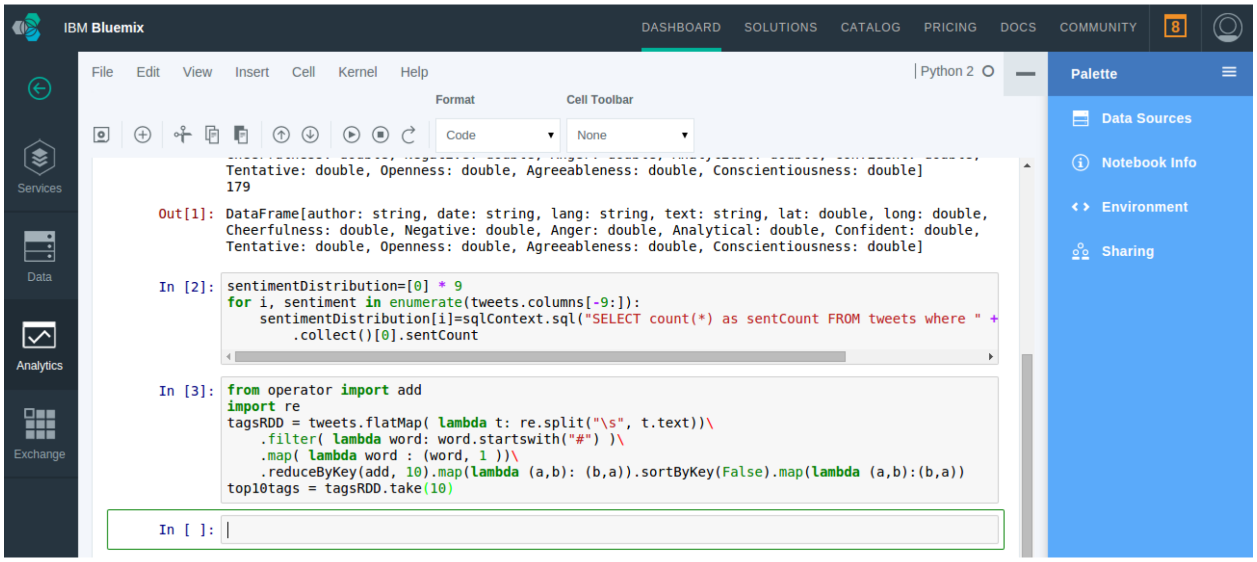
```
sentimentDistribution=[0] * 9
for i, sentiment in enumerate(tweets.columns[-9:]):
    sentimentDistribution[i]=sqlContext.sql("SELECT count(*) as sentCount FROM tweets where " + sentiment + " > 60")\
        .collect()[0].sentCount
```



```
        179
Out[1]: DataFrame[author: string, date: string, lang: string, text: string, lat: double, long: double,
        Cheerfulness: double, Negative: double, Anger: double, Analytical: double, Confident: double,
        Tentative: double, Openness: double, Agreeableness: double, Conscientiousness: double]

In [2]: sentimentDistribution=[0] * 9
        for i, sentiment in enumerate(tweets.columns[-9:]):
            sentimentDistribution[i]=sqlContext.sql("SELECT count(*) as sentCount FROM tweets where " +
                .collect()[0].sentCount

In [ ]:
```

Compute the top 10 hashtags contained in the tweets by using the following command,

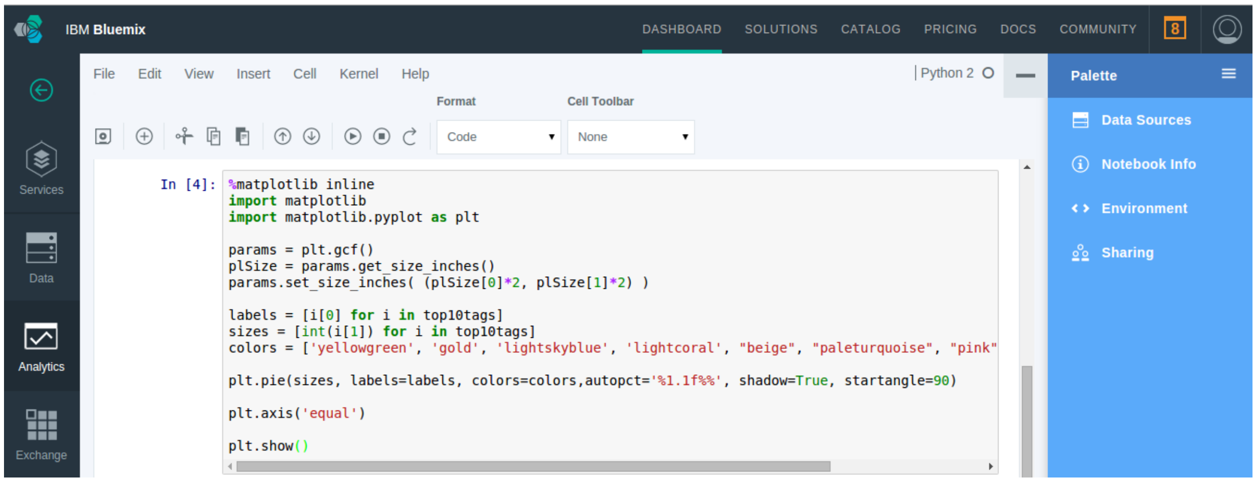
```
from operator import add
import re
tagsRDD = tweets.flatMap( lambda t: re.split("\s", t.text))\
    .filter( lambda word: word.startswith("#") )\
    .map( lambda word : (word, 1 ))\
    .reduceByKey(add, 10).map(lambda (a,b):
(b,a)).sortByKey(False).map(lambda (a,b):(b,a))
top10tags = tagsRDD.take(10)
```



```
        Tentative: double, Openness: double, Agreeableness: double, Conscientiousness: double]
        179

Out[1]: DataFrame[author: string, date: string, lang: string, text: string, lat: double, long: double,
        Cheerfulness: double, Negative: double, Anger: double, Analytical: double, Confident: double,
        Tentative: double, Openness: double, Agreeableness: double, Conscientiousness: double]

In [2]: sentimentDistribution=[0] * 9
        for i, sentiment in enumerate(tweets.columns[-9:]):
            sentimentDistribution[i]=sqlContext.sql("SELECT count(*) as sentCount FROM tweets where " +
                .collect()[0].sentCount

In [3]: from operator import add
        import re
        tagsRDD = tweets.flatMap( lambda t: re.split("\s", t.text))\
            .filter( lambda word: word.startswith("#") )\
            .map( lambda word : (word, 1 ))\
            .reduceByKey(add, 10).map(lambda (a,b): (b,a)).sortByKey(False).map(lambda (a,b):(b,a))
        top10tags = tagsRDD.take(10)

In [ ]:
```

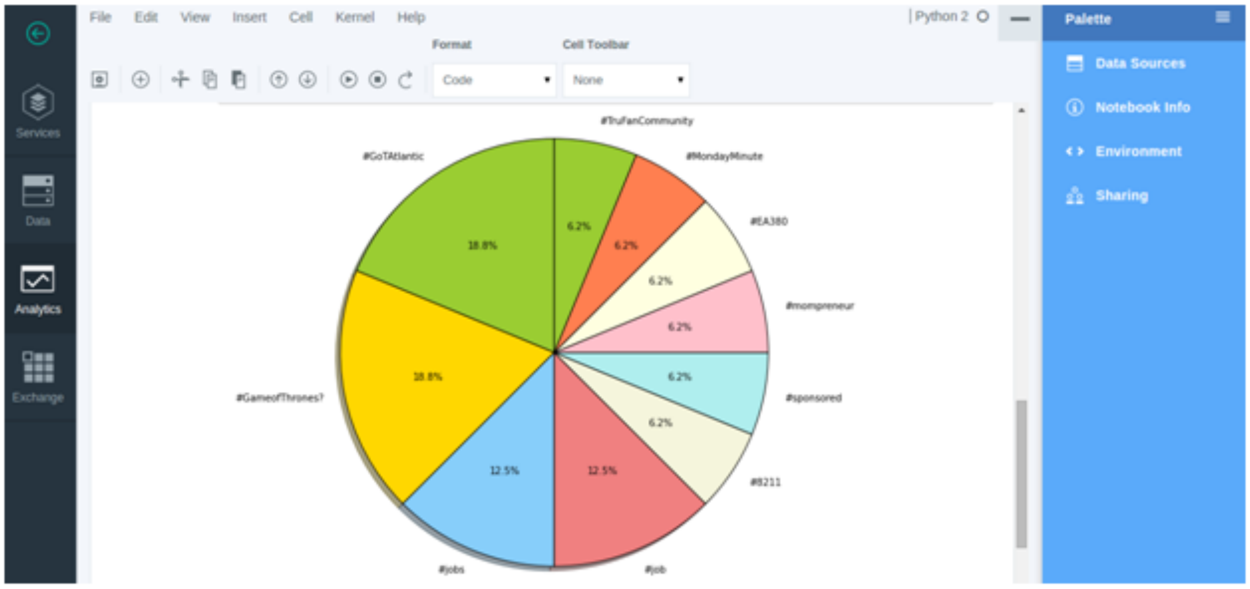
Plot the RDD as a pie chart by using the following code,

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
params = plt.gcf()
plSize = params.get_size_inches()
params.set_size_inches( (plSize[0]*2, plSize[1]*2) )
labels = [i[0] for i in top10tags]
sizes = [int(i[1]) for i in top10tags]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral',
"beige", "paleturquoise", "pink", "lightyellow", "coral"]
plt.pie(sizes, labels=labels, colors=colors,autopct='%1.1f%%',
shadow=True, startangle=90)
plt.axis('equal')
plt.show()
```



```
In [4]: %matplotlib inline
        import matplotlib
        import matplotlib.pyplot as plt

        params = plt.gcf()
        plSize = params.get_size_inches()
        params.set_size_inches( (plSize[0]*2, plSize[1]*2) )

        labels = [i[0] for i in top10tags]
        sizes = [int(i[1]) for i in top10tags]
        colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral', "beige", "paleturquoise", "pink"

        plt.pie(sizes, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True, startangle=90)

        plt.axis('equal')

        plt.show()
```

The top 10 Hashtags sentiment percentage will appear as a pie chart.


File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Format
Cell Toolbar
Code
None
Services
Data
Analytics
Exchange
Palette
Data Sources
Notebook Info
Environment
Sharing
#TruFanCommunity
#MondayMinute
#EA380
#mompreneur
#sponsored
#8211
#job
#jobs
#GameofThrones?
#GoTAtlantic