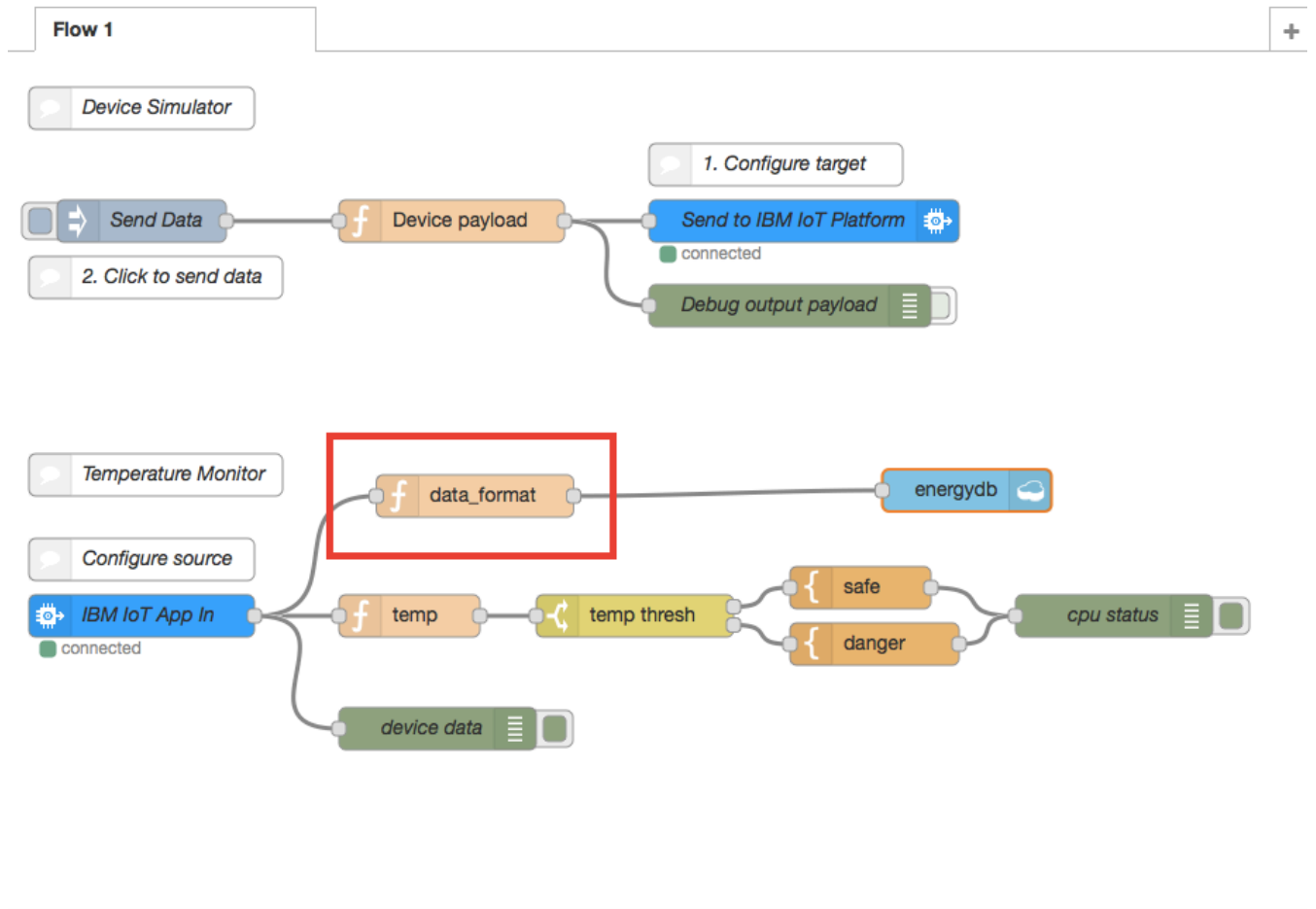


Data Visualization using Apache Spark Python Notebook



Step 1: Drag and drop function node from the left side in Node RED and wire it in between the IBM IoT Device and energydb as shown in the figure above.

Edit function node

Cancel

Done

Name

data_format



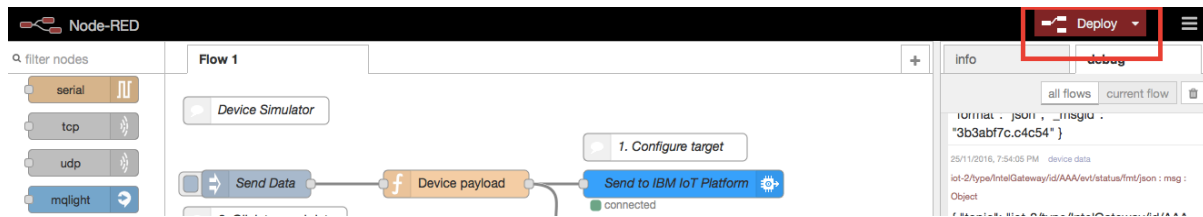
Function

```
1 msg={
2
3     "Voltage": msg.payload.d.Voltage,
4     "Frequency": msg.payload.d.Frequency,
5     "Kwh": msg.payload.d.Kwh,
6     "Time": msg.payload.d.Time,
7     "Current" : msg.payload.d.Current,
8     "Power_Factor" : msg.payload.d.Power_Factor,
9 }
10 return msg;
11
```

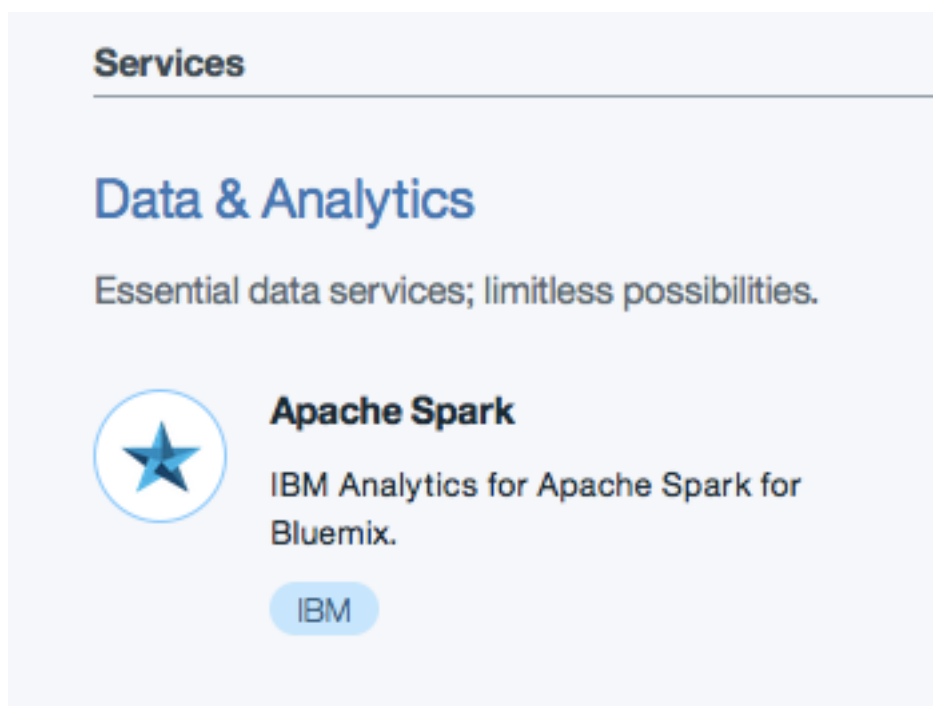
Outputs

1

Step 2: Write the same line of code in the function node as shown in the above image and click on **DONE**



Step 3: Click on **DEPLOY** as shown in the figure above and then go to Bluemix Catalog.



Step 4: From Catalog, in **Data & Analytics**, click on **Apache Spark**

IBM Bluemix Catalog

231CatalogSupportAccount

Apache Spark

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.

IBM

Connect to:

Leave unbound

Service name:

ApacheSpark

Credential name:

Credentials-1

Features

- 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.

Need Help?
[Contact Bluemix Sales](#)

Estimate Monthly Cost
[Cost Calculator](#)

Create

Step 5: Input the Service name and then click on **CREATE**

IBM Bluemix Data & Analytics

231CatalogSupportAccount

< All Items

ApacheSpark

Manage Service Credentials Connections

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.

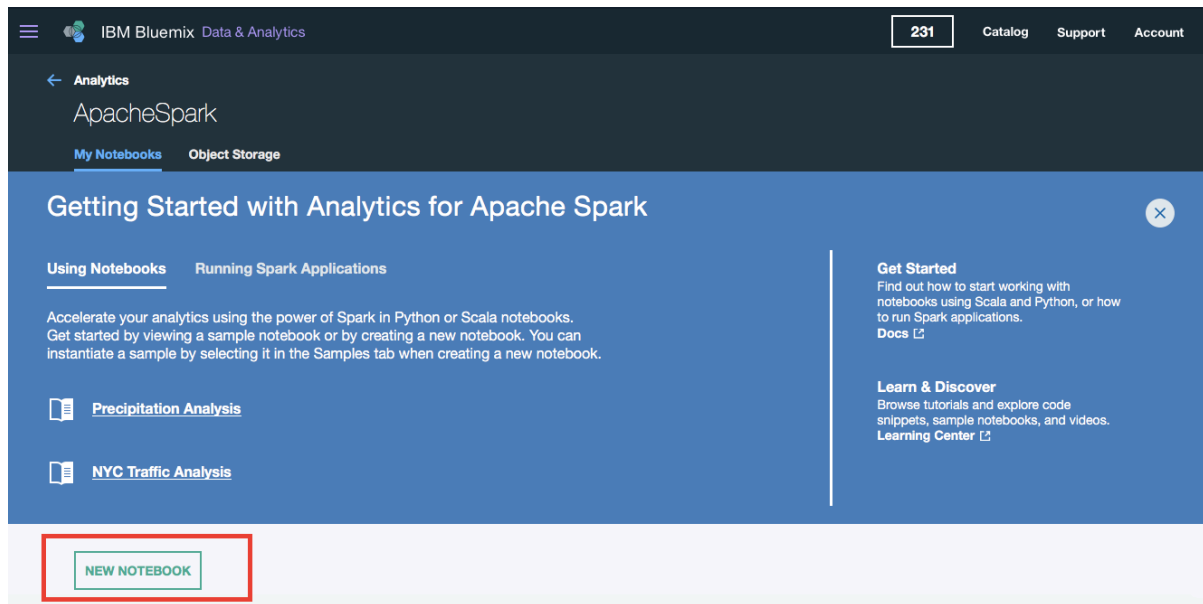
[Job History](#)

Learn

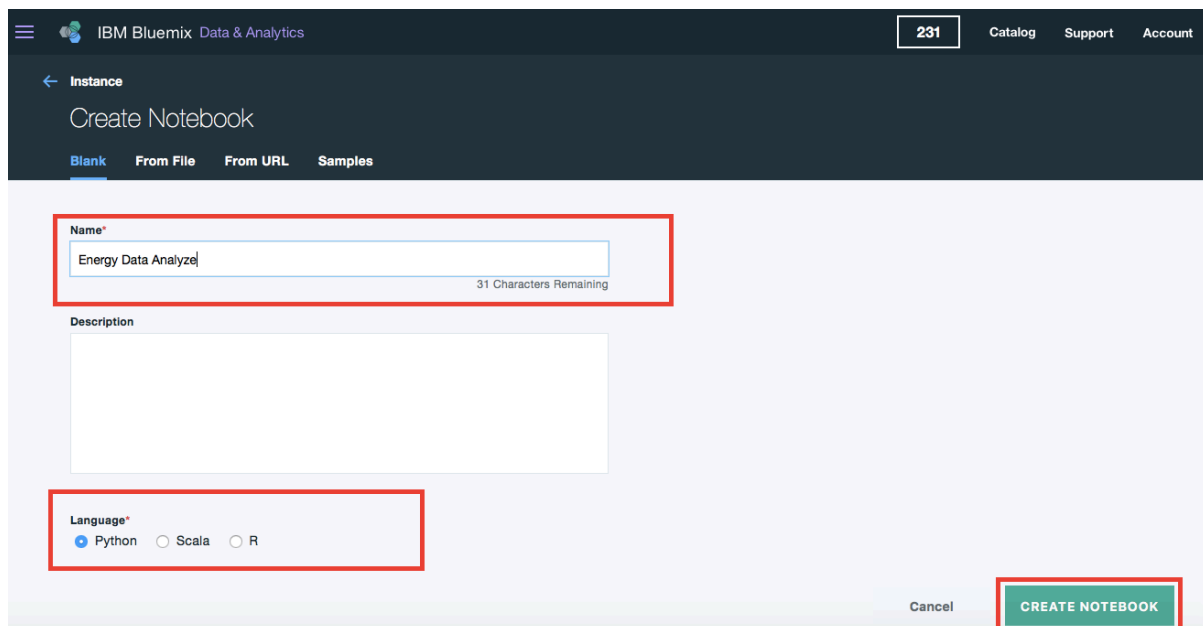
Find out how to get started with notebooks using Scala and Python, or how to run Spark applications.

[Docs](#)

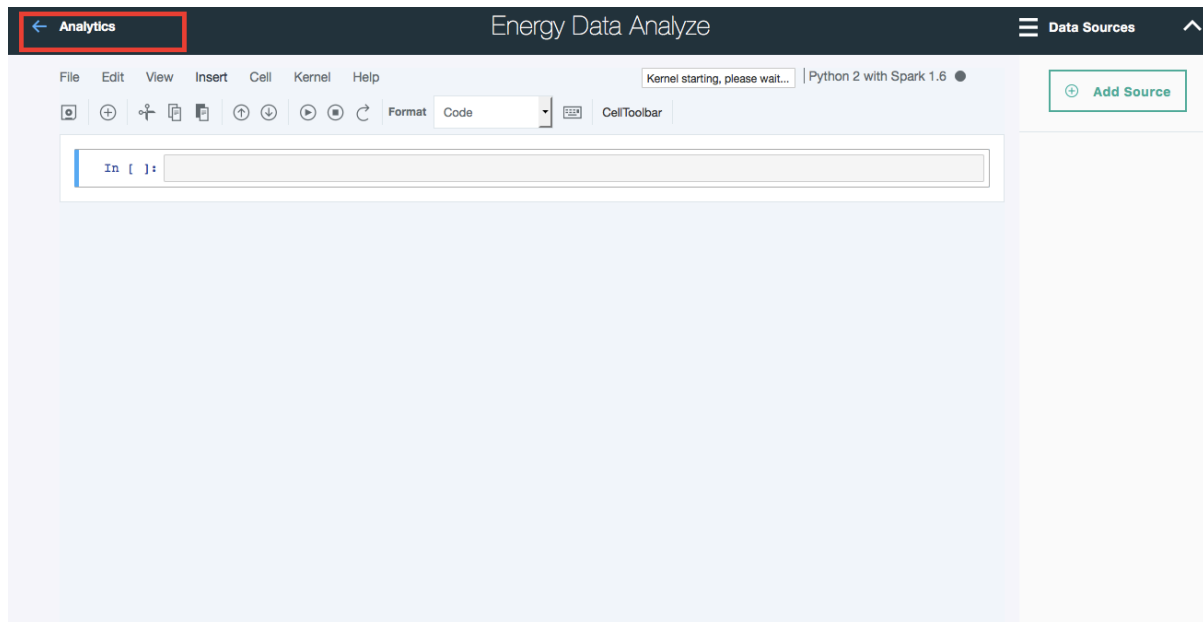
Step 6: Click on **NOTEBOOKS**



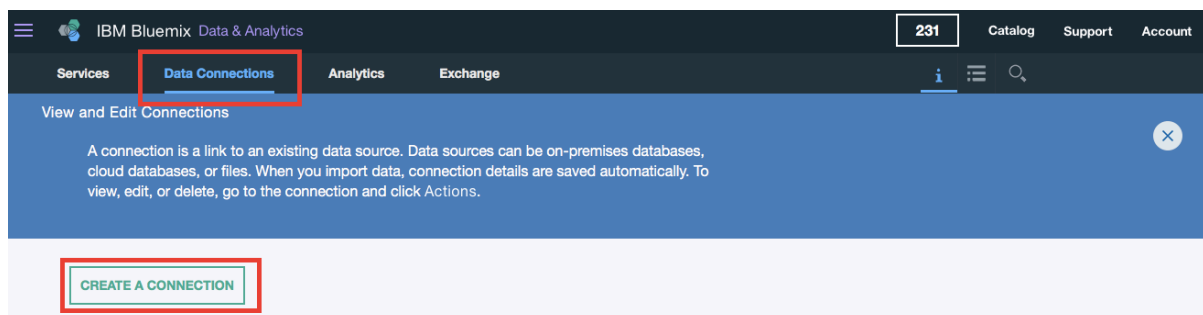
Step 7: In My Notebooks tab, Click on **NEW NOTEBOOK**



Step 8: Input the Name, Language as shown in the above image and then click on **CREATE NOTEBOOK**

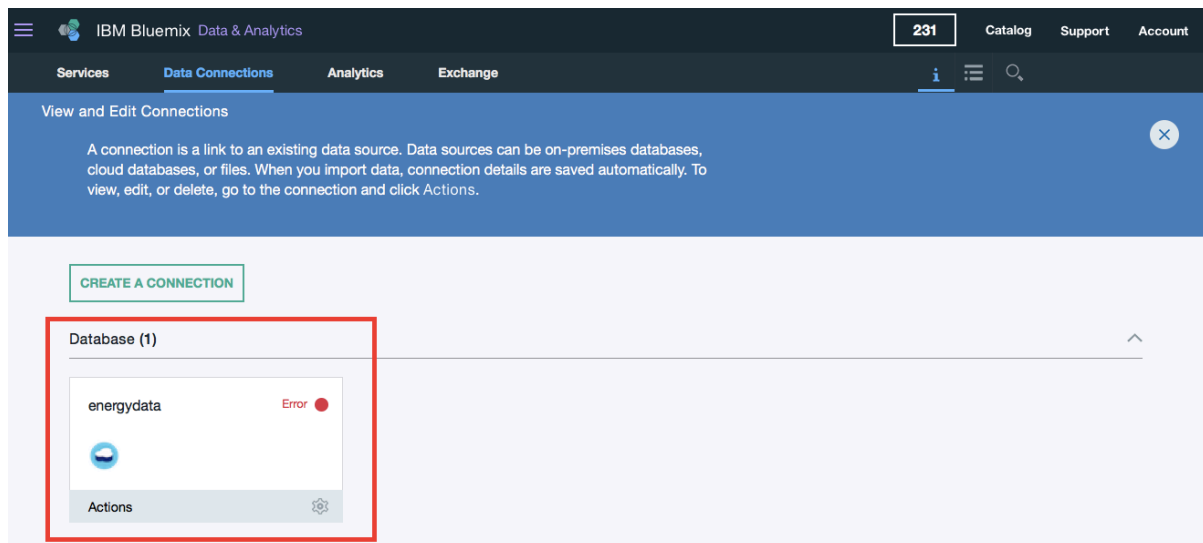


Step 9: Click on **ANALYTICS** as shown in the above image

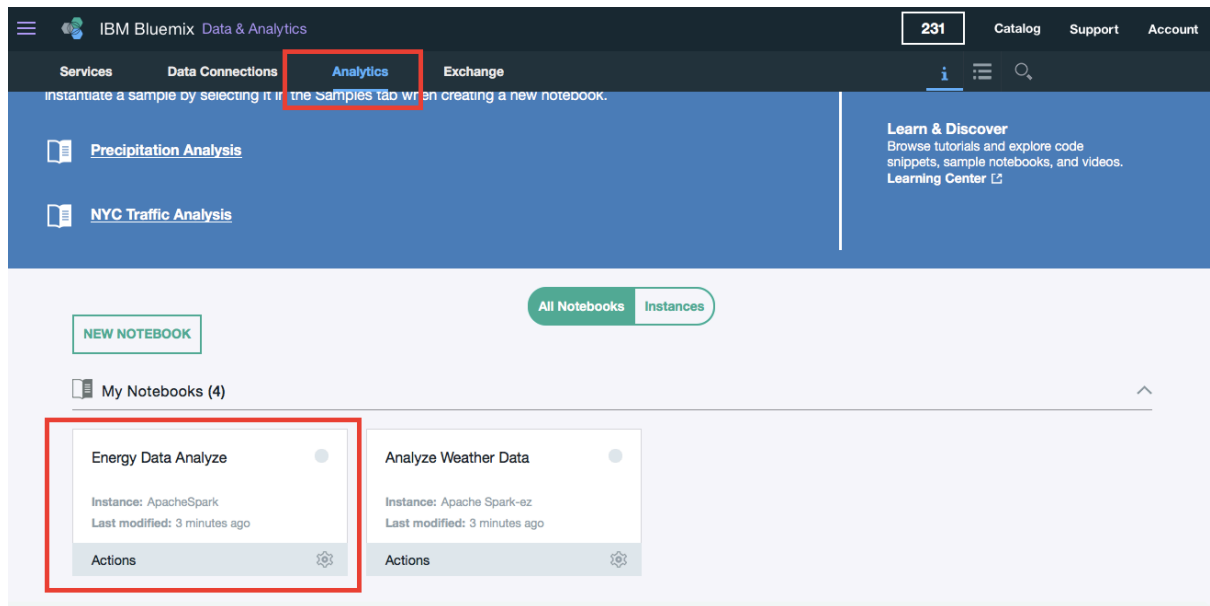


Step 10: Click on **DATA CONNECTIONS** and then click on **CREATE A CONNECTION** as shown in the above image

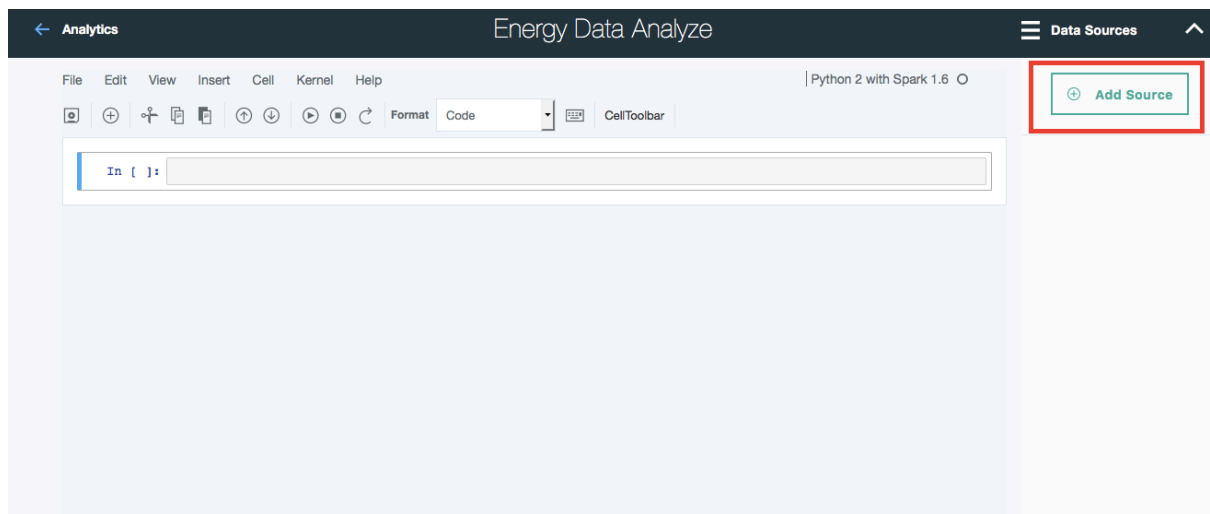
Step 11: Input Name and then Instance & Database name and Click on **CREATE CONNECTION** as shown in the image above



Step 12: Now you can see that your database has been added in the Data Connections Tab as shown in the image above



Step 13: Goto Analytics tab and click on the Energy Data Analysis as shown in the image above



Step 14: Click on **ADD SOURCE** as shown in the image above.



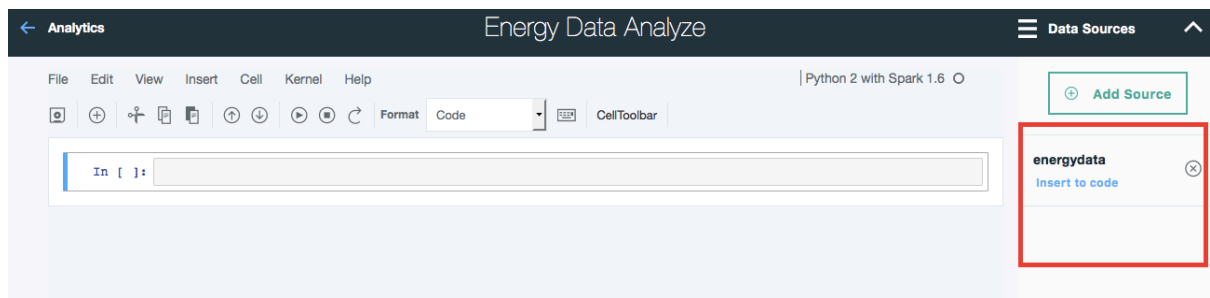
Add Data Source

| Name | Description | Select |
|------------|-------------|----------------------------------|
| energydata | | <input checked="" type="radio"/> |

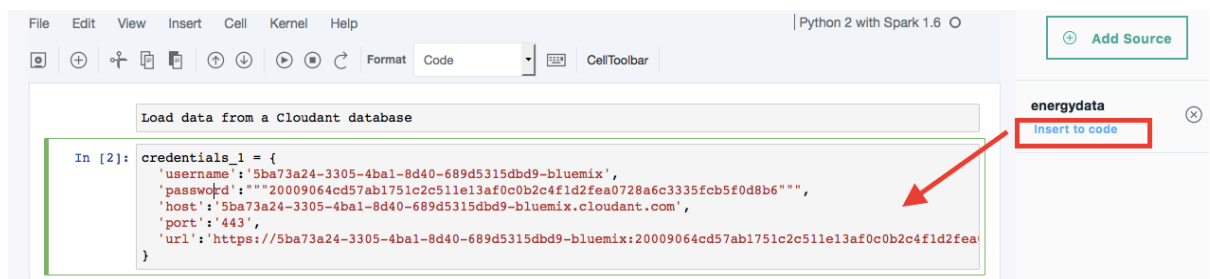
Selected data sources are referenced, not copied.
Create or connect to an [Object Storage](#) instance to conveniently upload data from a file for use in your notebooks.


CancelADD DATA SOURCE

Step 15: Click on the energydata radio button and then click on ADD DATA SOURCE



Step 16: Now you will see your database on the right side as shown in the image above.



Step 17: Add a cell by clicking on the + button (from menubar) and then **Insert to code** as shown in the image above. Then click on  button.


```

Install the cloudant package:

In [3]: !pip install --user cloudant

Collecting cloudant
  Downloading cloudant-2.3.0-py2-none-any.whl (63kB)
    100% |#####| 71kB 4.2MB/s
Requirement already satisfied (use --upgrade to upgrade): requests<3.0.0,>=2.7.0 in /usr/local/src/blue
mix_jupyter_bundle.v25/notebook/lib/python2.7/site-packages (from cloudant)
Installing collected packages: cloudant
Successfully installed cloudant-2.3.0

```

Step 18: Add another cell by clicking on + button(from menubar) and then add **!pip install --user cloudant** Then click on  button and result should be as shown in the image above

Connecting to Cloudant with the credentials

```
In [4]: from cloudant.client import Cloudant
        from cloudant.result import Result
        import pandas as pd, json

        client = Cloudant(credentials_1['username'], credentials_1['password'], url=credentials_1['url'])
        client.connect()
```


Step 19: Add another cell by clicking on + button(from menubar) and then add

lines of code as shown in the image above and Then click on  button.

List| all existing databases:

```
In [5]: client.all_dbs()
Out[5]: [u'_replicator', u'_users', u'energydb', u'nodered']
```

Step 20: Add another cell by clicking on + button(from menubar) and then add

`client.all_dbs()` Then click on  button. It will list out all the databases.


Listing documents in the Cloudant Database energydb

```
In [6]: db_name = 'energydb'
        my_database = client[db_name]
        result_collection = Result(my_database.all_docs, include_docs=True)
        data_df = pd.DataFrame([item['doc'] for item in result_collection])
        data_df.head()
```

```
Out[6]:
```

| | Current | Frequency | KwH | Power_Factor | Time | Voltage | _id | _rev |
|---|-----------|-----------|--------|--------------|---------------|------------|----------------------------------|-----------------|
| 0 | 10.003906 | 50 | 524288 | 0.9845 | 1480083098625 | 230.324768 | 063ffc18d986fb64557b64562317d1dc | 1-9125fc988206 |
| 1 | 10.003906 | 50 | 8192 | 0.9845 | 1480083134768 | 231.074768 | 063ffc18d986fb64557b6456231fee7b | 1-1c9d5581395 |
| 2 | 9.599854 | 50 | 128 | 0.9845 | 1480083143796 | 230.075012 | 063ffc18d986fb64557b64562321fee2 | 1-ef61ff62e745c |
| 3 | 10.003906 | 50 | 2048 | 0.9845 | 1480083167869 | 230.449768 | 063ffc18d986fb64557b64562326db35 | 1-ff4a87e1db72 |
| 4 | 10.003906 | 50 | 32768 | 0.9845 | 1480083207025 | 230.199768 | 063ffc18d986fb64557b6456232e1ddb | 1-a2b227744b9 |

Step 21: Add another cell by clicking on + button (from menubar) and then add

lines of code as shown in the image above. Then click on  button. You will get the output as shown in the image above.

Cleaning the data

```
In [7]: del data_df['_id']  
del data_df['_rev']
```

Step 22: Add another cell by clicking on + button (from menubar) and then add

lines of code as shown in the image above. Then click on  button.


Placing Time in X axis

```
In [8]: data_df = data_df.set_index(data_df["Time"])  
data_df.drop(['Time'], axis=1, inplace=True)  
data_df.head()
```

Out[8]:

| | Current | Frequency | KwH | Power_Factor | Voltage |
|---------------|-----------|-----------|--------|--------------|------------|
| Time | | | | | |
| 1480083098625 | 10.003906 | 50 | 524288 | 0.9845 | 230.324768 |
| 1480083134768 | 10.003906 | 50 | 8192 | 0.9845 | 231.074768 |
| 1480083143796 | 9.599854 | 50 | 128 | 0.9845 | 230.075012 |
| 1480083167869 | 10.003906 | 50 | 2048 | 0.9845 | 230.449768 |
| 1480083207025 | 10.003906 | 50 | 32768 | 0.9845 | 230.199768 |

Step 23: Add another cell by clicking on + button (from menubar) and then add

lines of code as shown in the image above. Then click on  button. You will get the output as shown in the image above

```
Including matplotlib
```

```
In [9]: %matplotlib inline
```

Step 24: Add another cell by clicking on + button(from menubar) and then add

lines of code as shown in the image above. Then click on  button.

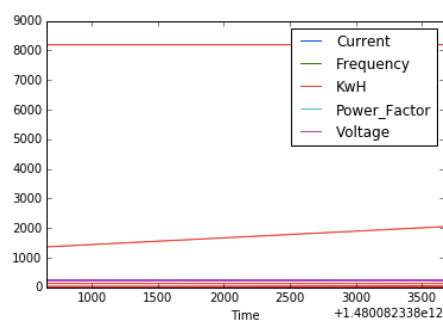
```
Converting String to Float datatype
```

```
In [10]: df=data_df.astype(float)
```


Step 25: Add another cell by clicking on + button(from menubar) and then add

lines of code as shown in the image above. Then click on  button.

```
In [68]: df.plot()  
Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x7f886534d290>
```



Step 26: Add another cell by clicking on + button(from menubar) and then add

lines of code as shown in the image above. Then click on  button. You should see the graph plotted and data visualized.