Data Import

How-To Guide

databricks™

Databricks Data Import How-To Guide

Databricks is an integrated workspace that lets you go from ingest to production, using a variety of data sources. Databricks is powered by Apache® Spark™, which can read from Amazon S3, MySQL, HDFS, Cassandra, etc. In this How-To Guide, we are focusing on S3, since it is very easy to work with. For more information about Amazon S3, please refer to Amazon Simple Storage Service (S3).

Loading data into S3

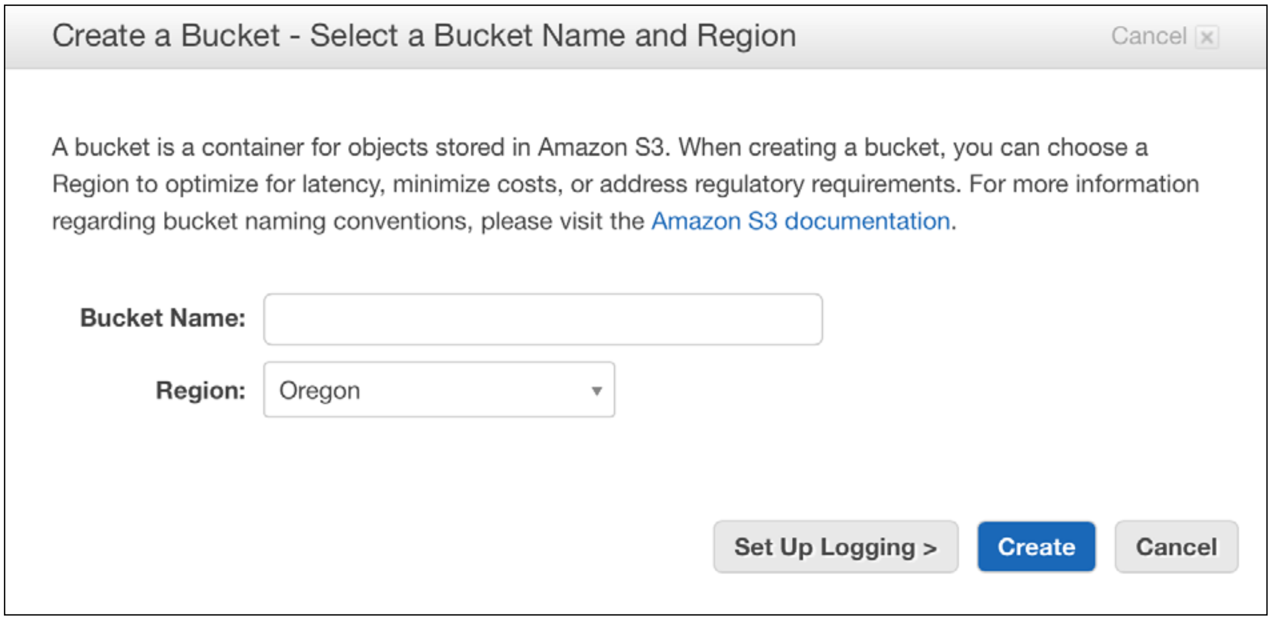
In this section, we describe two common methods to upload your files to S3. You can also reference the AWS documentation Uploading Objects into Amazon S3 or the AWS CLI s3 Reference.

Loading data using the AWS UI

For the details behind Amazon S3, including terminology and core concepts, please refer to the document What is Amazon S3. Below is a quick primer on how to upload data and presumes that you have already created your own Amazon AWS account.

1. Within your AWS Console, click on the S3 icon to access the S3 User Interface (it is under the **Storage & Content Delivery** section)

2. Click on the **Create Bucket** button to create a new bucket to store your data. Choose a unique name for your bucket and choose your region. If you have already created your Databricks account, ensure this bucket's region matches the region of your Databricks account. EC2 instances and S3 buckets should be in the same region to improve query performance and prevent any cross-region transfer costs.

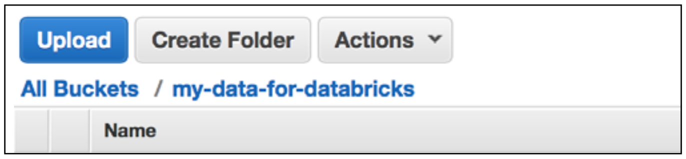

Create a Bucket - Select a Bucket Name and Region

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the Amazon S3 documentation.

Bucket Name:

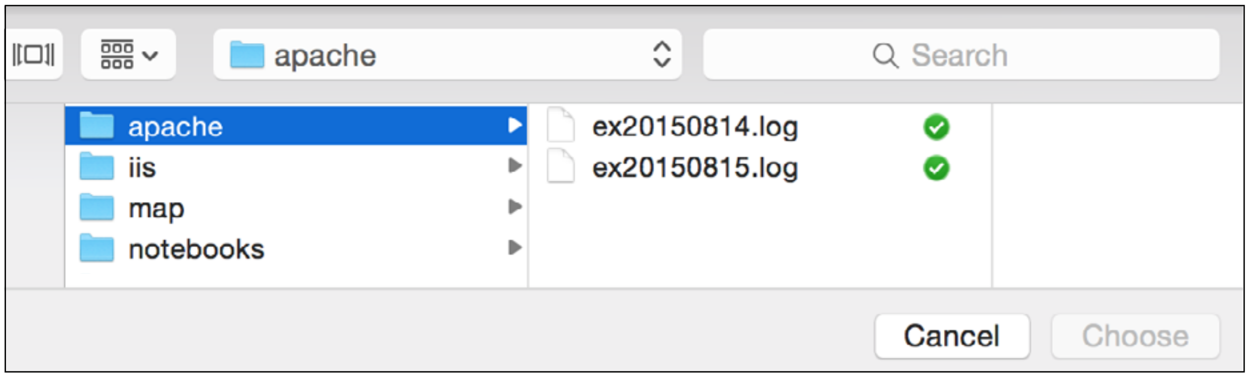
Region: Oregon

3. Click on the bucket you have just created. For the demonstration purposes, the name of my bucket is "my-data-for-databricks". From here, click on the **Upload** button.



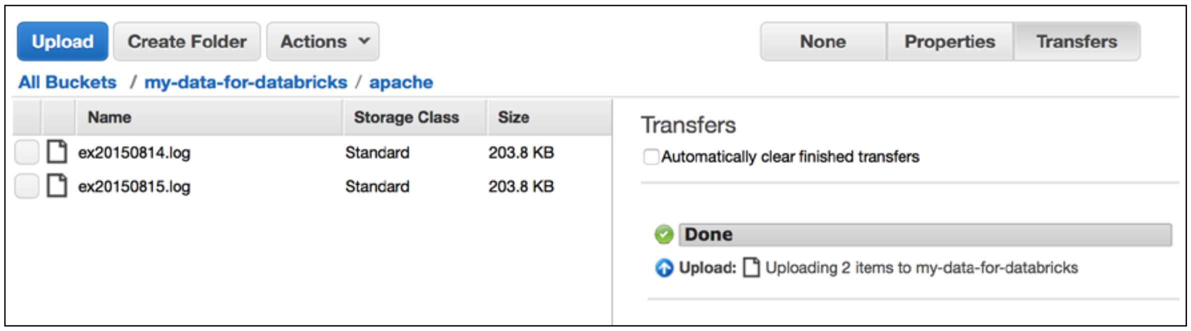
4. In the **Upload – Select Files and Folders** dialog, you will be able to add your files into S3.

5. Click on **Add Files** and you will be able to upload your data into S3. Below is the dialog to choose sample web logs from my local box.



Click **Choose** when you have selected your file(s) and then click **Start Upload**.

6. Once your files have been uploaded, the **Upload** dialog will show the files that have been uploaded into your bucket (in the left pane), as well as the transfer process (in the right pane).



Name	Storage Class	Size
ex20150814.log	Standard	203.8 KB
ex20150815.log	Standard	203.8 KB

Transfers

Automatically clear finished transfers

Done

Upload: Uploading 2 items to my-data-for-databricks

Now that you have uploaded data into Amazon S3, you are ready to use your Databricks account. Additional information:

- How to upload data using alternate methods, continue reading this document.
- How to connect your Databricks account to the data you just uploaded, please skip ahead to “Connecting to Databricks” on page 9.
- To learn more about Amazon S3, please refer to What is Amazon S3.

Loading data using the AWS CLI

If you are a fan of using a command line interface (CLI), you can quickly upload data into S3 using the AWS CLI. For more information including the reference guide and deep dive installation instructions, please refer to the AWS Command Line Interface page. These next few steps provide a high level overview of how to work with the AWS CLI.

Note, if you have already installed the AWS CLI and know your security credentials, you can skip to Step #3.

1. Install AWS CLI

    a) For Windows, please install the 64-bit or 32-bit Windows Installer (for most new systems, you would choose the 64-bit option).

    b) For Mac or Linux systems, ensure you are running Python 2.6.5 or higher (for most new systems, you would already have Python 2.7.2 installed) and install using pip.

    ```
    pip install awscli
    ```

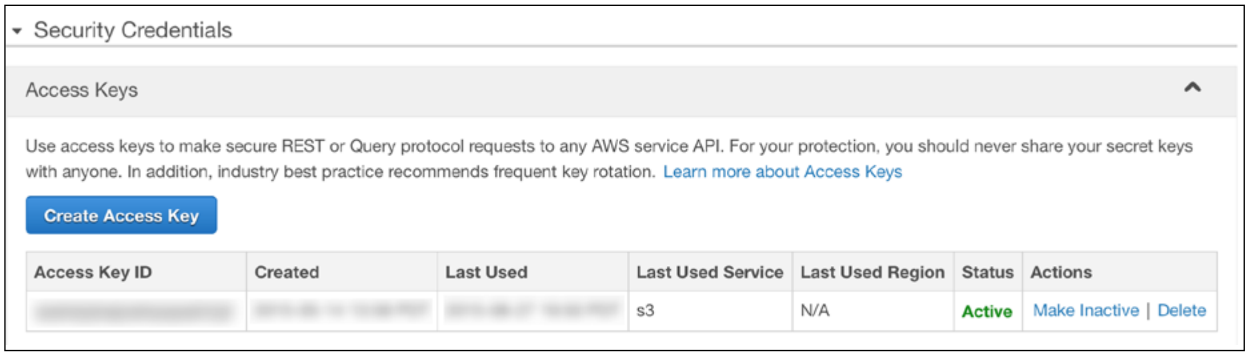
2. Obtain your AWS security credentials

To obtain your security credentials, log onto your AWS console and click on **Identity & Access Management** under the **Administration & Security** section. Then,

- Click on Users

- Find your own user name whom you will be using the user credentials
- Scroll down the menu to **Security Credentials > Access Keys**
- At this point, you can either **Create Access Key** or use an existing key if you already have one.

For more information, please refer to AWS security credentials.


Security Credentials

Access Keys

Use access keys to make secure REST or Query protocol requests to any AWS service API. For your protection, you should never share your secret keys with anyone. In addition, industry best practice recommends frequent key rotation. Learn more about Access Keys

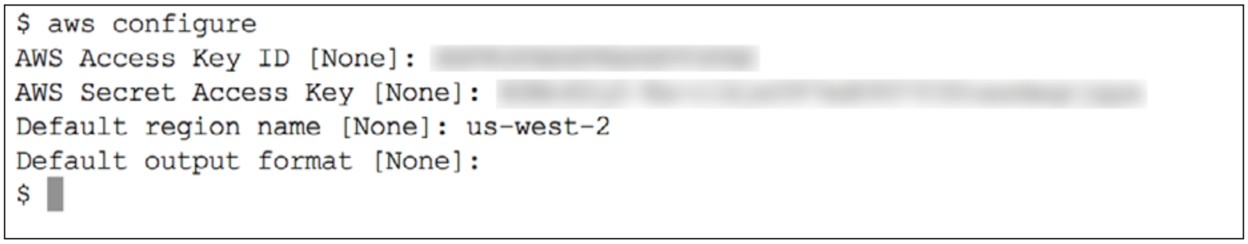
Create Access Key

Access Key ID	Created	Last Used	Last Used Service	Last Used Region	Status	Actions
[illegible]	[illegible]	[illegible]	s3	N/A	Active	Make Inactive | Delete

3. Configure your AWS CLI security credentials

```
aws configure
```

This command allows you to set your AWS security credentials (click for more information). When configuring your credentials, the resulting output should look something similar to the screenshot below.



```
$ aws configure
AWS Access Key ID [None]: [illegible]
AWS Secret Access Key [None]: [illegible]
Default region name [None]: us-west-2
Default output format [None]:
$
```

Note, the default region name is us-west-2 for the purpose of this demo. Based on your geography, your default region name may be different. You can get the full listing of S3 region-specific end points at Region and End Points > Amazon Simple Storage Service (S3).

4. Copy your files to S3

Create a bucket for your files (for this demo, the bucket being created is "my-data-for-databricks") using the make bucket (**mb**) command.

```
    aws s3 mb s3://my-data-for-databricks/
```

Then, you can copy your files up to S3 using the copy (**cp**) command.

```
    aws s3 cp . s3://my-data-for-databricks/ --recursive
```

If you would like to use the sample logs that are used in this technical note, you can download the log files from http://bit.ly/1MuGbJy.

The output of from a successful copy command should be similar the one below.

```
upload: ./ex20111215.log to s3://my-data-for-
databricks/ex20111215.log
upload: ./ex20111214.log to s3://my-data-for-
databricks/ex20111214.log
```

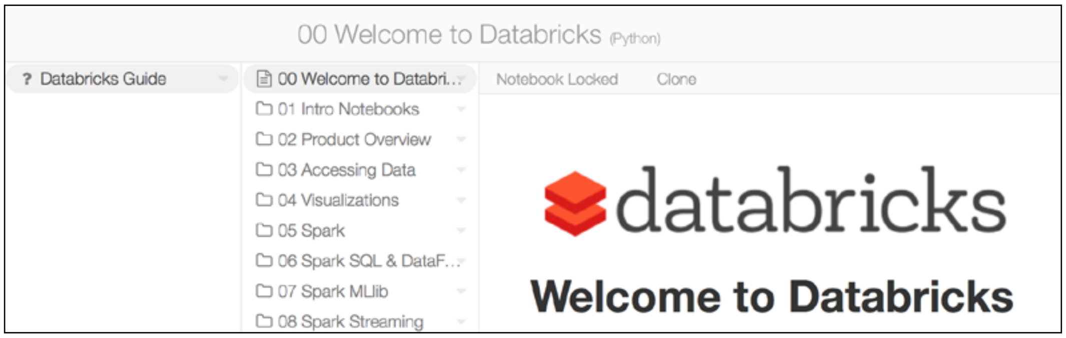
Connecting to Databricks

In the previous section, we covered the steps required to upload your data into S3. In this section, we will cover how you can access this data within Databricks. This section presumes the following

- You have completed the previous section and/or have AWS credentials to access data.
- You have a Databricks account; if you need one, please go to Databricks Account for more information.
- You have a running Databricks cluster.

For more information, please refer to:

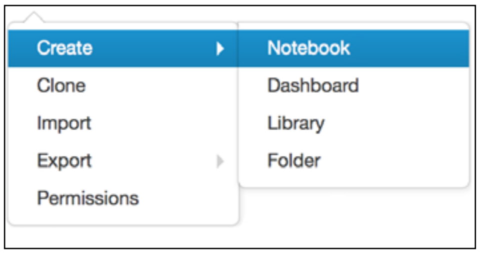
- Introduction to Databricks video
- **Welcome to Databricks** notebook in the Databricks Guide (top item under Workspace when you log into your Databricks account).



Accessing your Data from S3

For this section, we will be connecting to S3 using Python referencing the Databricks Guide notebook **03 Accessing Data > 2 AWS S3 – py**. If you want to run these commands in Scala, please reference the **03 Accessing Data > 2 AWS S3 – scala** notebook.

1. Create a new notebook by opening the main menu ≡, click on down arrow on the right side of **Workspace**, and choose **Create > Notebook**



2. Mount your S3 bucket to the Databricks File System (DBFS).

   This allows you to avoid entering AWS keys every time you connect to S3 to access your data (i.e. you only have to enter the keys once). A DBFS mount is a pointer to S3 and allows you to access the data as if your files were stored locally.

```
import urllib
ACCESS_KEY = "REPLACE_WITH_YOUR_ACCESS_KEY"
SECRET_KEY = "REPLACE_WITH_YOUR_SECRET_KEY"
ENCODED_SECRET_KEY = urllib.quote(SECRET_KEY, "")
AWS_BUCKET_NAME = "REPLACE_WITH_YOUR_S3_BUCKET"
MOUNT_NAME = "REPLACE_WITH_YOUR_MOUNT_NAME"

dbutils.fs.mount("s3n://%s:%s@%s" % (ACCESS_KEY, ENCODED_
SECRET_KEY, AWS_BUCKET_NAME), "/mnt/%s" % MOUNT_NAME)
```

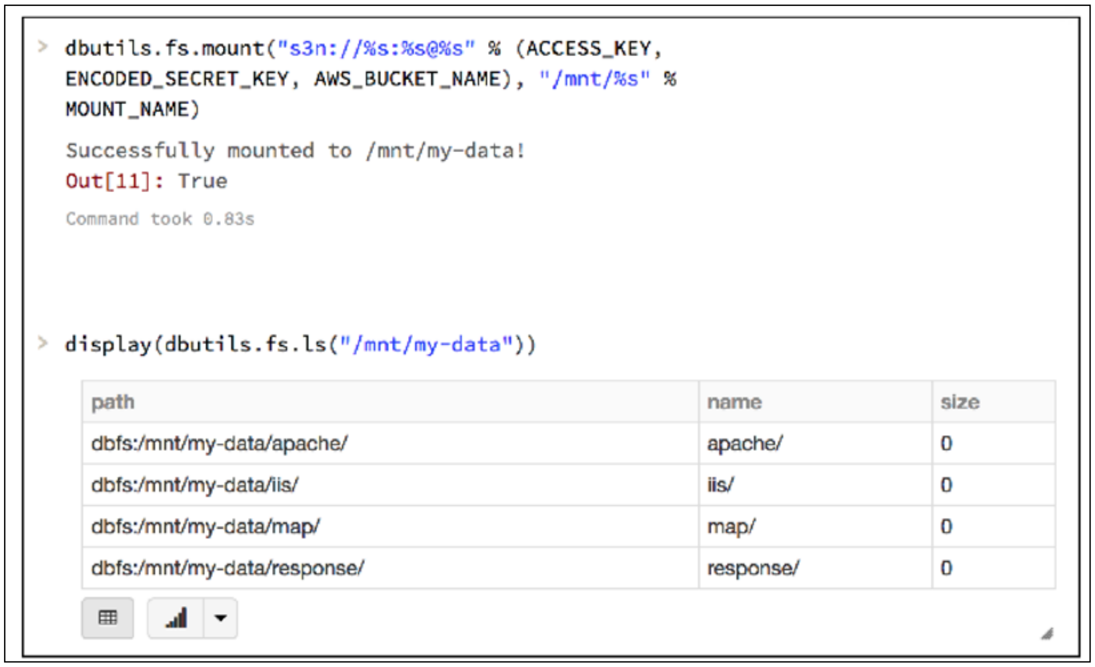
Remember to replace the "REPLACE_WITH…" statements prior to executing the command.Once you have mounted the bucket, delete the above cell so others do not have access to those keys.

3. Once you've mounted your S3 bucket to DBFS, you can access it by using the command below. It is accessing dbutils, available in Scala and Python, so you can run file operations command from your notebooks.

```
display(dbutils.fs.ls("/mnt/my-data"))
```

Note: in the previous command, the "REPLACE_WITH_YOUR_MOUNT_NAME" statement with the value "my-data" hence the folder name is /mnt/my-data.

Below is example output for running these commands within a Python notebook.



```
> dbutils.fs.mount("s3n://%s:%s@%s" % (ACCESS_KEY,
  ENCODED_SECRET_KEY, AWS_BUCKET_NAME), "/mnt/%s" %
  MOUNT_NAME)

  Successfully mounted to /mnt/my-data!
  Out[11]: True

  Command took 0.83s
```

```
> display(dbutils.fs.ls("/mnt/my-data"))
```

path	name	size
dbfs:/mnt/my-data/apache/	apache/	0
dbfs:/mnt/my-data/iis/	iis/	0
dbfs:/mnt/my-data/map/	map/	0
dbfs:/mnt/my-data/response/	response/	0

Querying Data from the DBFS mount

In the previous section, you had created a DBFS mount point allowing you to connect to your S3 location as if it were a local drive without the need to re-enter your S3 credentials.

Querying from Python RDD

From the same notebook, you can now run the commands below to do a simple count against your web logs.

```
myApacheLogs = sc.textFile("/mnt/my-data/")
myApacheLogs.count()
```

The output from this command should be similar to the output below.

```
> myApacheLogs.count()
  Out[18]: 4468
       Command took 0.98s
```

Query tables via SQL

While you can read these weblogs using a Python RDD, we can quickly convert this to DataFrame accessible by Python and SQL. The following commands convert the `myApacheLogs` RDD into a DataFrame.

```
# sc is an existing SparkContext.
from pyspark.sql import SQLContext, Row

# Load the space-delimited web logs (text files)
parts = myApacheLogs.map(lambda l: l.split(" "))
apachelogs = parts.map(lambda p: Row(ipaddress=p[0],
clientidentd=p[1], userid=p[2], datetime=p[3], tmz=p[4],
method=p[5], endpoint=p[6], protocol=p[7], responseCode=p[8],
contentSize=p[9]))

# Infer the schema, and register the DataFrame as a table.
schemaWebLogs = sqlContext.createDataFrame(apachelogs)
schemaWebLogs.registerTempTable("apachelogs")
```

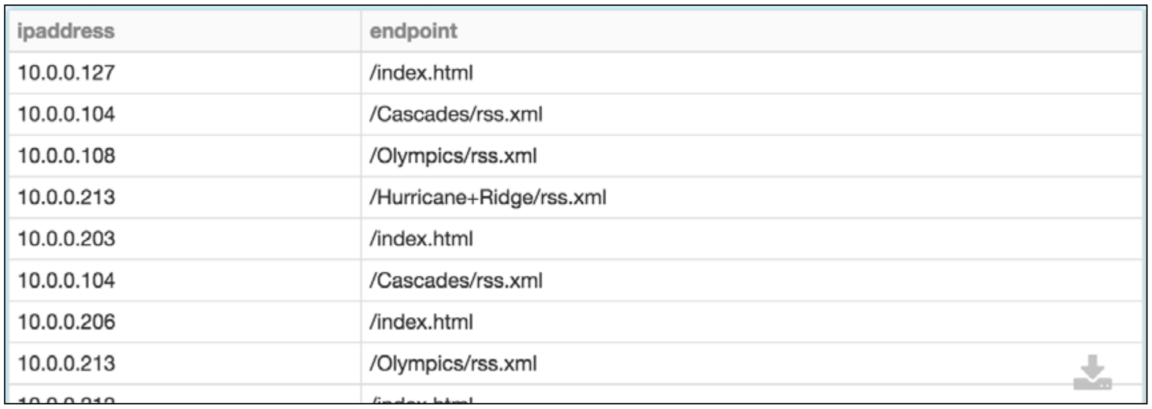
After running these commands successfully from your Databricks python notebook, you can run SQL commands over your apachelogs DataFrames that has been registered as a table. The output should be similar to the one below.

```
sqlContext.sql("select ipaddress, endpoint from apachelogs").take(10)

Out[23]:
[Row(ipaddress=u'10.0.0.127', endpoint=u'/index.html'),
 Row(ipaddress=u'10.0.0.104', endpoint=u'/Cascades/rss.xml'),
 Row(ipaddress=u'10.0.0.108', endpoint=u'/Olympics/rss.xml'),
 Row(ipaddress=u'10.0.0.213', endpoint=u'/Hurricane+Ridge/rss.xml'),
 Row(ipaddress=u'10.0.0.203', endpoint=u'/index.html'),
 Row(ipaddress=u'10.0.0.104', endpoint=u'/Cascades/rss.xml'),
 Row(ipaddress=u'10.0.0.206', endpoint=u'/index.html'),
 Row(ipaddress=u'10.0.0.213', endpoint=u'/Olympics/rss.xml'),
 Row(ipaddress=u'10.0.0.212', endpoint=u'/index.html'),
 Row(ipaddress=u'10.0.0.114', endpoint=u'/index.html')]
```

Because you had registered the weblog DataFrame, you can also access this directly from a Databricks SQL notebook. For example, below is screenshot of running the SQL command

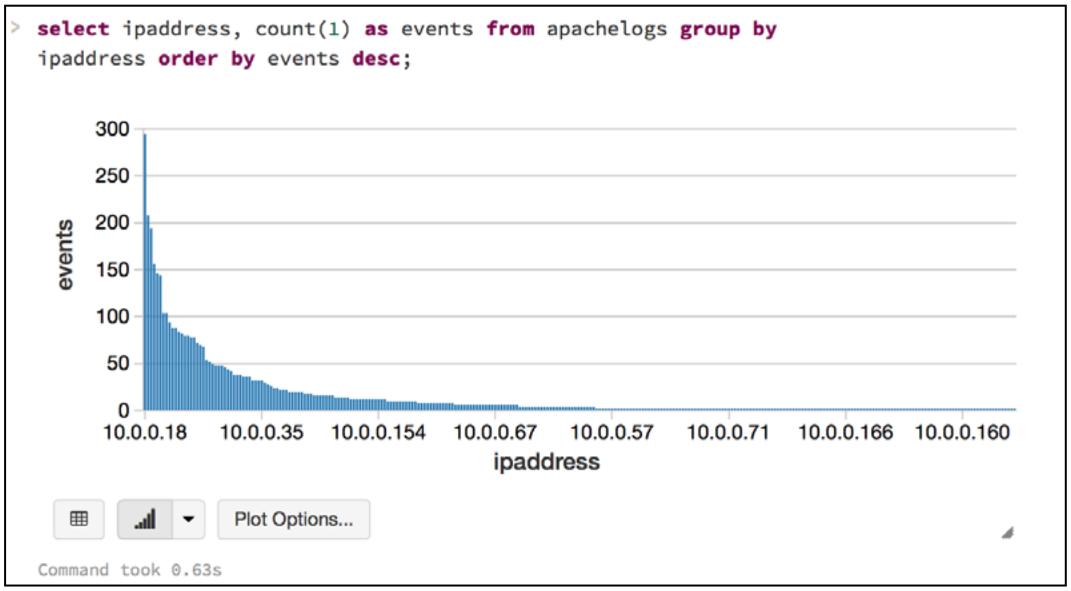
```
select ipaddress, endpoint from weblogs limit 10;
```



ipaddress	endpoint
10.0.0.127	/index.html
10.0.0.104	/Cascades/rss.xml
10.0.0.108	/Olympics/rss.xml
10.0.0.213	/Hurricane+Ridge/rss.xml
10.0.0.203	/index.html
10.0.0.104	/Cascades/rss.xml
10.0.0.206	/index.html
10.0.0.213	/Olympics/rss.xml

With the query below, you can start working with the notebook graphs.

```
select ipaddress, count(1) as events from apachelogs
group by ipaddress order by events desc;
```



```
> select ipaddress, count(1) as events from apachelogs group by
  ipaddress order by events desc;
```

Command took 0.63s

Summary

This How-To Guide has provided a quick jump start on how to import your data into AWS S3 as well as into Databricks.

For next steps, please continue with:

- Continue the Introduction sections of the Databricks Guide
- Review the Log Analysis Example: How-to Guide.
- Watch a Databricks Webinar including
  - Building a Turbo-fast Data Warehousing Platform with Databricks
  - Apache Spark DataFrames: Simple and Fast Analysis of Structured Data.

Additional Resoures

If you'd like to analyze your Apache access logs with Databricks, you can evaluate Databricks with a trial account now. You can also find the source code on Github.

Other Databricks how-tos can be found at:
The Easiest Way to Run Spark Jobs

Evaluate Databricks with a trial account now:

databricks.com/try-databricks