Analyzing Data with Spark in Azure Databricks

Lab 3 – Using Structured Streaming

Overview

In this lab, you will run a Spark job to continually process a real-time stream of data.

What You'll Need

To complete the labs, you will need the following:

- A web browser
- A Microsoft account
- A Microsoft Azure subscription
- A Windows, Linux, or Mac OS X computer
- Azure Storage Explorer
- The lab files for this course

Note: To set up the required environment for the lab, follow the instructions in the **Setup** document for this course. Specifically, you must have signed up for an Azure subscription.

Provisioning Azure Resources

Note: If you already have an Azure Databricks Spark cluster and an Azure blob storage account, you can skip this section.

Provision a Databricks Workspace

1. In a web browser, navigate to http://portal.azure.com, and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Microsoft Azure portal, click **+ Create a resource**. Then in the **Analytics** section select **Azure Databricks** and create a new Azure Databricks workspace with the following settings:
    - **Workspace name**: *Enter a unique name (and make a note of it!)*
    - **Subscription:** *Select your Azure subscription*
    - **Resource Group:** *Create a new resource group with a unique name (and make a note of it!)*
    - **Location:** *Choose any available data center location.*
    - **Pricing Tier:** Standard
3. In the Azure portal, view **Notifications** to verify that deployment has started. Then wait for the workspace to be deployed (this can take few minutes).

Provision a Storage Account

1. In the Azure portal tab in your browser, and click **+ Create a resource**.
2. In the **Storage** category, click **Storage account**.
3. Create a new storage account with the following settings:
    - **Name**: *Specify a unique name (and make a note of it)*
    - **Deployment model**: Resource manager
    - **Account kind**: Storage (general purpose v1)
    - **Location**: *Choose the same location as your Databricks workspace*
    - **Replication:** Locally-redundant storage (LRD)
    - **Performance:** Standard
    - **Secure transfer required:** Disabled
    - **Subscription:** *Choose your Azure subscription*
    - **Resource group:** *Choose the existing resource group for your Databricks workspace*
    - **Virtual networks:** Disabled
4. Wait for the resource to be deployed. Then view the newly deployed storage account.
5. In the blade for your storage account, click **Blobs**.
6. In the **Browse blobs** blade, click **+ Container**, and create a new container with the following settings:
    - **Name**: spark
    - **Public access level**: Private (no anonymous access)
7. In the **Settings** section of the blade for your blob store, click **Access keys** and note the **Storage account name** and **key1** values on this blade – you will need these in the next procedure.

Create a Spark Cluster

1. In the Azure portal, browse to the Databricks workspace you created earlier, and click **Launch Workspace** to open it in a new browser tab.
2. In the Azure Databricks workspace home page, under **New**, click **Cluster**.
3. In the **Create Cluster** page, create a new cluster with the following settings:
    - **Cluster Mode**: Standard
    - **Cluster Name**: *Enter a unique cluster name (and make a note of it)*
    - **Databricks Runtime Version**: *Choose the latest available version*
    - **Python Version:** 3
    - **Driver Type**: Same as worker
    - **Worker Type**: *Leave the default type*
    - **Min Workers:** 1
    - **Max Workers:** 2
    - **Auto Termination:** Terminate after 60 minutes.
    - **Spark Config**: Add two key-value pairs for your storage account and key like this:

      fs.azure.account.key.***your_storage_account***.blob.core.windows.net   ***your_key1_value***

      spark.hadoop.fs.azure.account.key.***your_storage_account***.blob.core.windows.net ***your_key1_value***

      Note: The first setting enables code using the newer Dataframe-based API to access your storage account. The second setting is used by the older RDD-based API.
4. Wait for the cluster to be created.

Process a Stream of Data

Spark structured streaming enables you to use the dataframe API to read and process an unbounded stream of data. This kind of processing is used in real-time scenarios to aggregate data over temporal intervals or *windows*. You can use Spark to process streaming data from a wide range of sources, including Azure Event Hubs, Kafka, and others. In this lab, you'll process data as it is added to a folder in Azure blob storage.

Upload Initial Data to Azure Storage

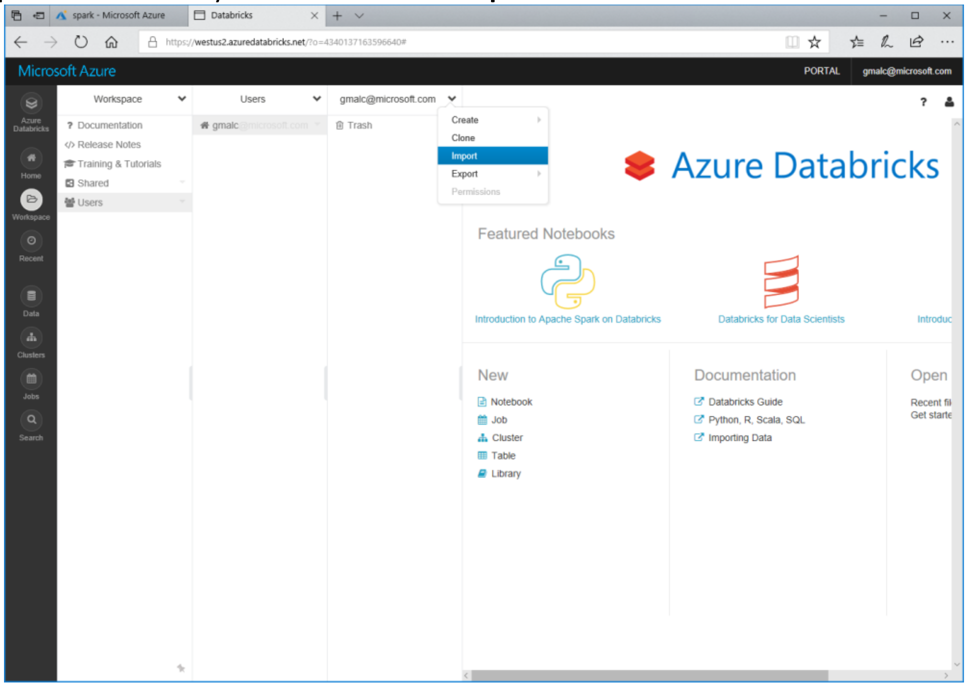
In this exercise, you will process a stream of data that simulates status information generated by Internet-of-things (IoT) devices. The data will be written to a blob storage container where it can be accessed by your Spark cluster. The instructions here assume you will use Azure Storage Explorer to upload the data, but you can use any Azure Storage tool you prefer.

1. In the folder where you extracted the lab files for this course on your local computer, verify that the **data\stream** folder contains four files named **stream_*N*.txt**. These files contain simulated status data from hypothetical IoT devices.
2. Start Azure Storage Explorer, and if you are not already signed in, sign into your Azure subscription.
3. Expand your storage account and the **Blob Containers** folder, and then double-click the **spark** blob container you created previously.
4. In the **Upload** drop-down list, click **Upload Files**. Then upload <u>only</u> **stream_1.txt** as a block blob to a new folder named **stream** in root of the **spark** container.

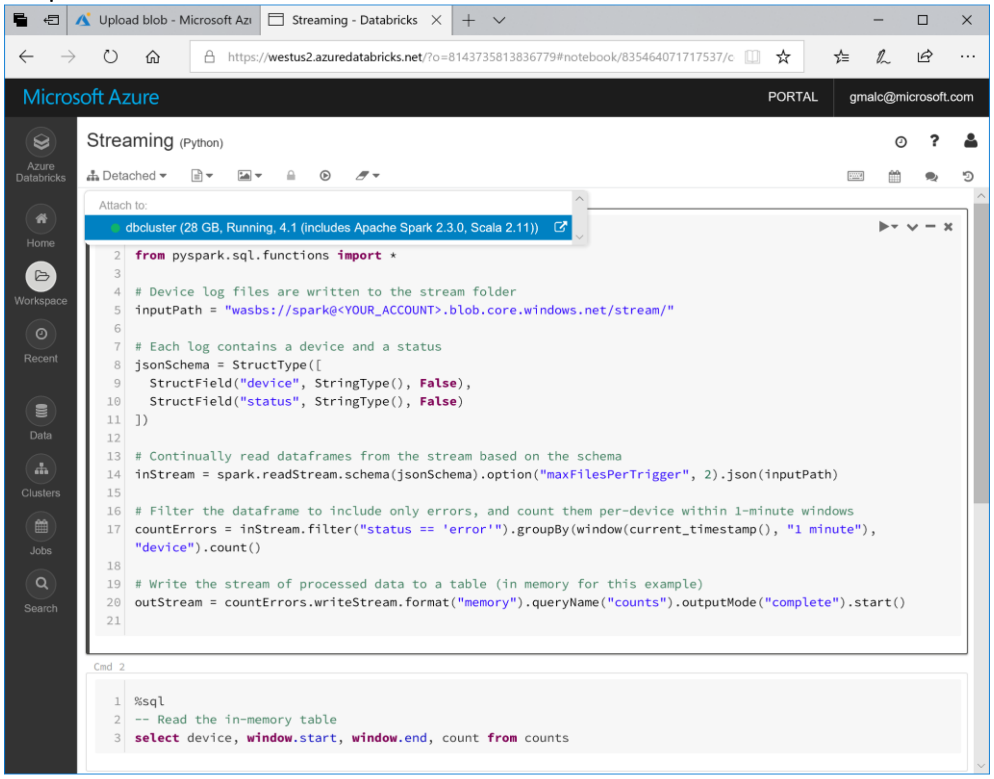
Upload Code to Process the Data Stream

In this lab, you will use your choice of a Python or Scala script to process the streaming IoT data. Source files containing the necessary code to process the data have been provided.

1. In the Databricks workspace, click **Workspace**. Then click **Users**, click your user name, and in the drop-down menu for your username click **Import** as shown here:



2. Browse to the folder where you extracted the lab files. Then select either **Streaming.py** or **Streaming.scala**, depending on your preferred choice of language (Python or Scala), and upload it.
3. Open the file you uploaded to view the code it contains, and read the comments to understand what it does.
4. Open the notebook you uploaded and in the **Detached** drop-down menu, attach the notebook to your Spark cluster as shown here:


Upload blob - Microsoft Azı
Streaming - Databricks
https://westus2.azuredatabricks.net/?o=8143735813836779#notebook/835464071717537/c
Microsoft Azure
PORTAL
gmalc@microsoft.com
Azure Databricks
Home
Workspace
Recent
Data
Clusters
Jobs
Search
Streaming (Python)
Detached
Attach to:
dbcluster (28 GB, Running, 4.1 (includes Apache Spark 2.3.0, Scala 2.11))
Cmd 2

```
from pyspark.sql.functions import *

# Device log files are written to the stream folder
inputPath = "wasbs://spark@<YOUR_ACCOUNT>.blob.core.windows.net/stream/"

# Each log contains a device and a status
jsonSchema = StructType([
  StructField("device", StringType(), False),
  StructField("status", StringType(), False)
])

# Continually read dataframes from the stream based on the schema
inStream = spark.readStream.schema(jsonSchema).option("maxFilesPerTrigger", 2).json(inputPath)

# Filter the dataframe to include only errors, and count them per-device within 1-minute windows
countErrors = inStream.filter("status == 'error'").groupBy(window(current_timestamp(), "1 minute"),
"device").count()

# Write the stream of processed data to a table (in memory for this example)
outStream = countErrors.writeStream.format("memory").queryName("counts").outputMode("complete").start()

```

```
%sql
-- Read the in-memory table
select device, window.start, window.end, count from counts
```

Process the Streaming Data

Now that you've uploaded the stream processing code, you're ready to use it to capture real-time device status and aggregate it over temporal windows.

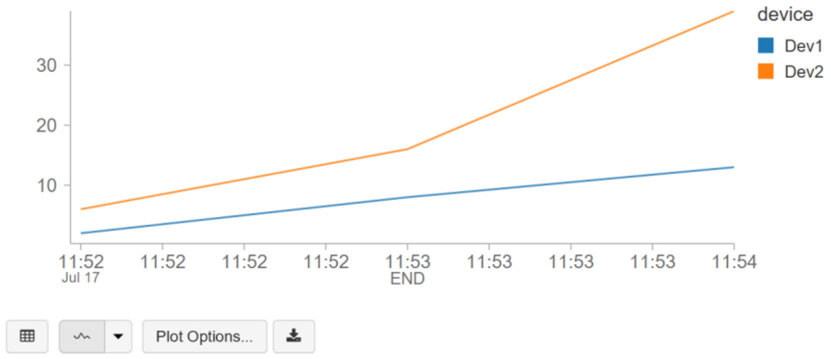
1. In the **Streaming** notebook, modify the first cell to replace **YOUR_ACCOUNT** with the name of your Azure storage account. The run the cell to start a streaming context and continually read data from the **stream** folder.
2. Wait for the stream to initialize and the **counts** streaming query to be created and then run the code in the second cell, which queries the in-memory table of streaming data. If there are no results, wait for a few seconds and try again.
3. While the stream processing code is still running, use your preferred Azure Storage tool to upload the **stream_2.txt** file to the **stream** folder in your **spark** blob container.
4. Return to the notebook and run the second cell again to verify that the new data added to the folder is included in the stream.
5. Wait 30 seconds or so and then repeat the previous two steps to upload **stream_3.txt** and **stream_4.txt**; running the second cell in the notebook after each upload to see the new data in the stream.

Visualize the Data

Your streaming query is now continually capturing data as files are added to the blob storage location, and aggregating the data values over 1-minute windows. You can plot this data to see trends in the data over time.

1. Under the table of device error data, in the plot drop-down list. click **Line** to plot the data as a line chart.
2. Click **Plot Options** and configure the following settings:
    - **Keys**: End
    - **Series groupings:** Device
    - **Values:** Count
    - **Aggregation:** Sum

    The plot should now show the total errors for each device as counted at the end of each 1-minute window, like this:



Stop the Streaming Query

When you are finished capturing data, you can stop the streaming query.

1. Run the code in the third code cell to stop the query.

Clean Up

Note: If you intend to proceed straight to the next lab, skip this section. Otherwise, follow the steps below to delete your Azure resources and avoid being charged for them when you are not using them.

Delete the Resource Group

1. Close the browser tab containing the databricks workspace if it is open.
2. In the Azure portal, view your **Resource groups** and select the resource group you created for your databricks workspace. This resource group contains your databricks workspace and your storage account.
3. In the blade for your resource group, click **Delete**. When prompted to confirm the deletion, enter the resource group name and click **Delete**.
4. Wait for a notification that your resource group has been deleted.
5. After a few minutes, a second resource group containing the resources for your cluster will automatically be deleted.
6. Close the browser.