FY19 Intro to Databricks Hack – Proctor Guide

Introduction to using Spark on Azure Databricks

# Goals

Most challenges observed by customers in these realms are in stitching multiple services together. As such, where possible, we have tried to place key concepts in the context of a broader example.

Once all hackathon challenges are completed, you should be able to:

* Create an Azure Databricks cluster
* Create Azure Databricks workspace and notebook for data exploration
* Understand the difference between Spark 1 and Spark 2 (RDD API vs. Dataframe API)
* How to use Spark SQL API to wrangle datasets using a familiar syntax

# Background Knowledge

This hackathon is meant for data engineers & data scientists on Azure. Previous experience with the following may be helpful but is not required:

* + Spark
  + Azure HD Insight
  + Python
  + Jupyter notebooks

# Introduction

In this challenge, you will provision a Databricks workspace and a Spark cluster. You will then use the Spark

cluster to explore data interactively.

Getting Started:

To complete the challenges, 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 challenge files for this course

**Note**: To set up the required environment for the challenge, follow the instructions in the [**Setup**](https://github.com/MicrosoftLearning/databricks-intro/raw/master/Setup%20Guide.pdf) 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.

Challenge 1 (details for proctor only / not for attendees – see end of document for what attendees will see):

Exploring Data Interactively with Spark RDDs

Now that you have provisioned a Spark cluster, you can use it to analyze data. In this exercise, you will

use Spark Resilient Distributed Datasets (RDDs) to load and explore data. The RDD-based API is an

original component of Spark, and has largely been superseded by a newer Dataframe-based API;

however, there are many production systems (and code examples on the Web) that use RDDs, so it’s

worth starting your exploration of Spark there.

Upload Source Data to Azure Storage

In this exercise, you will use the Spark RDD API to explore unstructured data. Before you can do this, you

must store the data files you want to explore in a blob storage container where it can be accessed by

your cluster. The instructions here assume you will use Azure Storage Explorer to do this, but you can

use any Azure Storage tool you prefer.

1. In the folder where you extracted the challenge files for this course on your local computer, in the **data**

folder, verify that the **KennedyInaugural.txt** file exist. This file contains the data you will explore

in this exercise.

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 **KennedyInaugural.txt** as a block

blob to a new folder named **data** in root of the **spark** container.

Create a Notebook

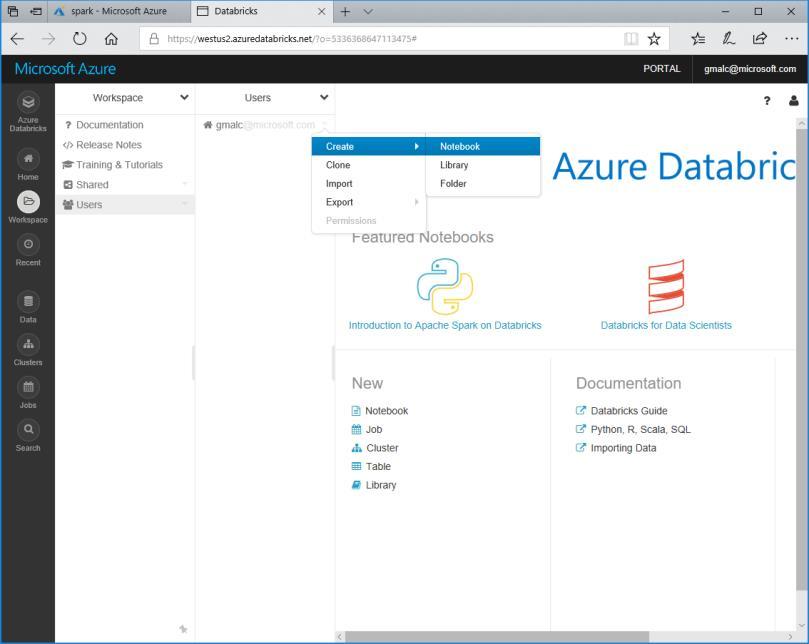
Most interactive data analysis in Databricks is conducted using *notebooks*. These browser-based

interactive documents enable you to combine notes in Markdown format with code that you can run

right in the notebook – with no requirement to install a local code editing environment. In this exercise,

you can choose to write your code using Python or Scala.

1. In the Databricks workspace, click **Workspace**. Then click **Users**, click your user name, and in the

drop-down menu for your username point click **Create** and **Notebook** as shown here:

2. Create a new notebook with the following settings:

• **Name**: RDDs

• **Language**: *Choose* Python *or* Scala *as preferred*.

• **Cluster**: *Your cluster*

3. In the new notebook, in the first cell, enter the following code to enter some Markdown text:

%md

# Kennedy Inauguration

This notebook contains code to analyze President Kennedy’s inauguration speech.

4. Click anywhere in the notebook outside of the first cell to see the formatted markdown, which

should look like this:

**Kennedy Inauguration**

This notebook contains code to analyze President Kennedy’s inauguration speech.

5. Hold the mouse pointer under the center of the bottom edge of the cell until a **(+)** symbol is

displayed; then click this to insert a new cell.

6. In the new cell, type the following code, replacing ***<account>*** with the fully qualified name of

your Azure Storage account (***account\_name*.blob.core.windows.net**):

*Python*

txt = sc.textFile("wasbs://spark@***<account>***/data/KennedyInaugural.txt")

txt.count()

*Scala*

val txt = sc.textFile("wasbs://spark@***<account>***/data/KennedyInaugural.txt")

txt.count()

In this code, the variable **sc** is the Spark context for your cluster; which is created automatically

within the notebook.

7. With the code cell selected, at the top left of the cell, click the  button and then click  **Run**

**Cell** to run the cell. After a few seconds, the code will run and display the number of lines of text

in the text file as **Out[1]**

8. Add a new cell and enter the following command to view the first line in the text file.

*Python*

txt.first()

*Scala*

txt.first()

9. Run the new cell and note that the first line of the speech is displayed as **Out[2]**.

10. Add a new cell and enter the following command to create a new RDD named **filtTxt** that filters

the **txt** RDD so that only lines containing the word “freedom” are included, and counts the

filtered lines.

*Python*

filtTxt = txt.filter(lambda line: "freedom" in line)

filtTxt.count()

*Scala*

val filtTxt = txt.filter(line => line.contains("freedom"))

filtTxt.count()

11. Run the new cell and note that the number of lines containing “freedom” is returned as **Out[3]**.

12. Add a new cell and enter the following command to display the contents of the **filtTxt** RDD.

*Python*

filtTxt.collect()

*Scala*

filtTxt.collect()

13. Run the new cell and note that the lines containing “freedom” are returned as **Out[4]**.

14. Add a new cell and enter the following command to split the full speech into words, count the

number of times each word occurs, and display the counted words in descending order of

frequency.

*Python*

words = txt.flatMap(lambda txt: txt.split(" "))

counts = words.map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)

counts.sortBy(lambda a: a[1], False).collect()

*Scala*

val words = txt.flatMap(line => line.split(" "))

val counts = words.map(word => (word, 1)).reduceByKey((a, b) => a + b)

counts.sortBy(\_.\_2,false).collect().foreach(println)

15. Run the new cell and review the output, which shows the frequency of each word in the speech

in descending order.

Exploring Data Interactively with Dataframes

Spark 2.0 and later provides a schematized object for manipulating and querying data – the DataFrame.

This provides a much more intuitive, and better performing, API for working with structured data. In

addition to the native Dataframe API, Spark SQL enables you to use SQL semantics to create and query

tables based on Dataframes.

Upload Source Data to Azure Storage

In this challenge, you will explore both structured data relating to road traffic accidents. Before you can do

this, you must upload the data files containing the data to your blob storage container where it can be

accessed by your cluster. The instructions here assume you will use Azure Storage Explorer to do this,

but you can use any Azure Storage tool you prefer.

1. In the folder where you extracted the challenge files for this course on your local computer, in the **data** folder, verify that the **Accidents.csv** and **Vehicles.csv** files exist. These files contain the data you

will explore in this exercise.

2. Start Azure Storage Explorer, and if you are not already signed in, sign into your Azure

subscription.

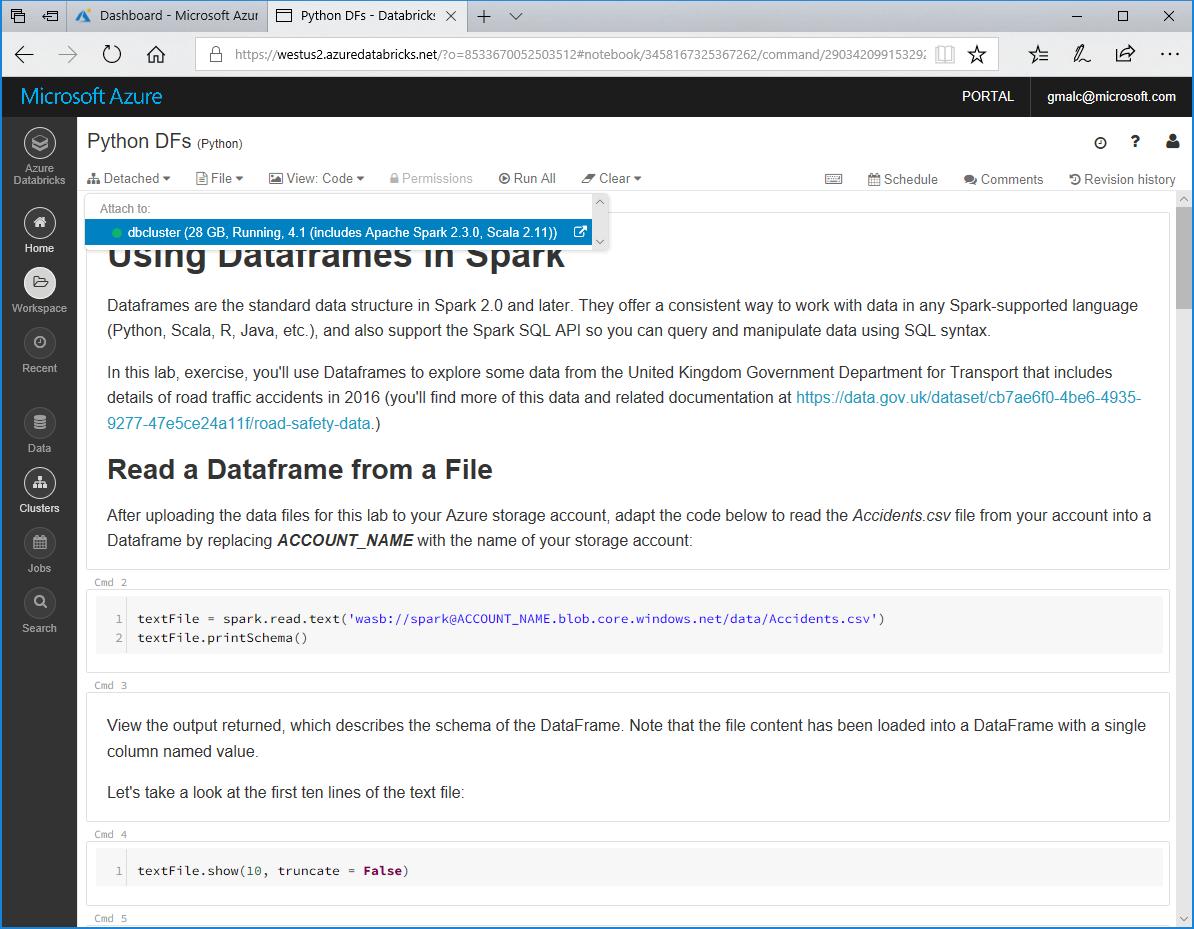
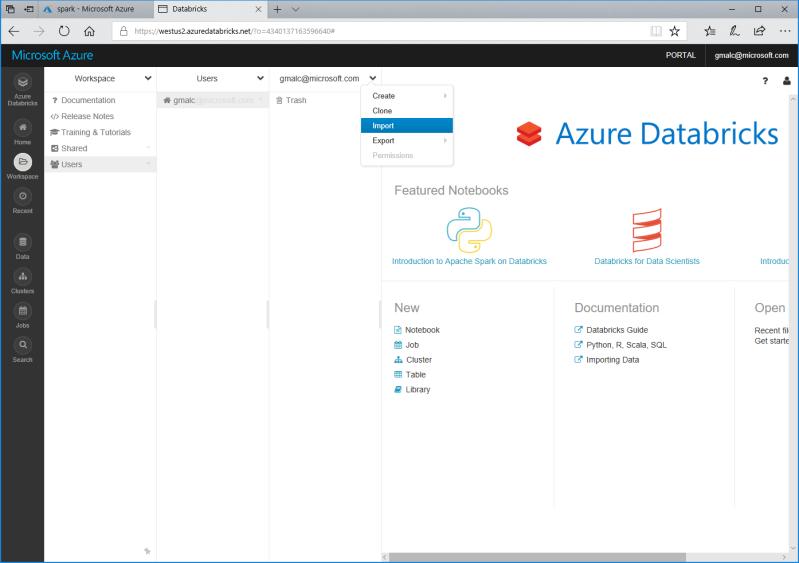
3. Expand your storage account and the **Blob Containers** folder, and then double-click the **spark**

blob container you created previously in this challenge.

4. In the **Upload** drop-down list, click **Upload Files**. Then upload **Accidents.csv** and **Vehicles.csv** as

block blobs to the **data** folder in root of the **spark** container that you created previously in this

challenge.



Work with Dataframes

In this procedure, you will use your choice of Python or Scala to query the road traffic accident data

in the comma-delimited text files you have uploaded. Notebooks containing the necessary steps to

explore 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 challenge files. Then select either **Dataframes.ipynb** or

**Dataframes.scala**, depending on your preferred choice of language (Python or Scala), and

upload it.

3. Open the notebook you uploaded and in the **Detached** drop-down menu, attach the notebook

to your Spark cluster as shown here:

4. Read the notes and run the code cells to explore the data.

Clean Up

**Note**: If you intend to proceed straight to the next challenge, 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.

Challenge 2: Creating a Spark Job

Spark jobs enable you to run data processing code on-demand or at scheduled intervals. This enables

you to build data processing solutions for unattended execution.

Upload Source Data to Azure Storage

In this exercise, you will create a Spark job to process a web server log file. Before you can do this, you

must store the log file you want to process in a blob storage container where it can be accessed by your

cluster. The instructions here assume you will use Azure Storage Explorer to do this, but you can use any

Azure Storage tool you prefer.

1. In the folder where you extracted the challenge files for this course on your local computer, in the **data**

folder, verify that the **IISlog.txt** file exist. This file contains the data you will process in this

exercise.

2. Open **IISlog.txt** in a text editor and view its contents. It contains details of page requests made

to a website, including requests to a page named *products.aspx* for which a query string

indicates the specific product that the user is viewing. Close the file without saving any changes.

3. Start Azure Storage Explorer, and if you are not already signed in, sign into your Azure

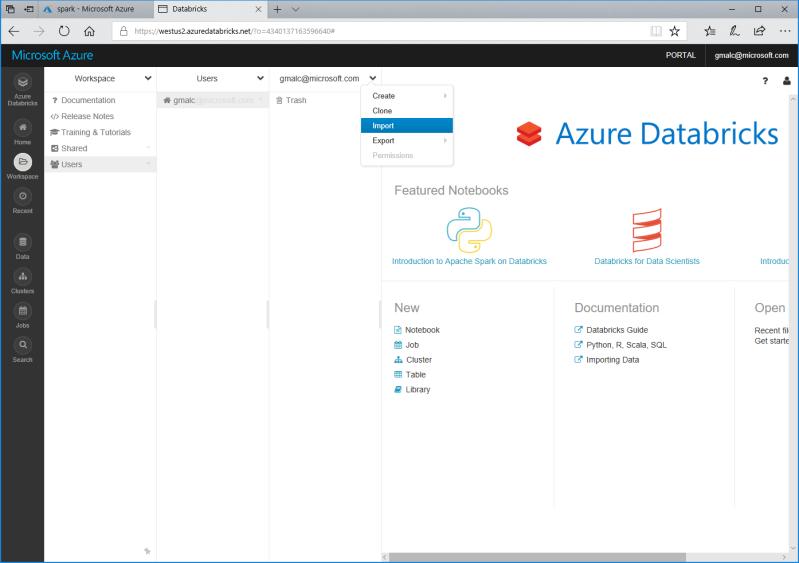
subscription.

4. Expand your storage account and the **Blob Containers** folder, and then double-click the **spark**

blob container you created previously.

5. In the **Upload** drop-down list, click **Upload Files**. Then upload **IISlog.txt** as a block blob to a

folder named **data** in root of the **spark** container.



Upload Code to Process the Data

In this challenge, you will use your choice of a Python or Scala script to process the web server log data you

have uploaded. 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 challenge files. Then select either **ProcessLog.py** or

**ProcessLog.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, read the comments to understand what

it does, and change both instances of *<YOUR\_ACCOUNT>* to the name of your Azure Storage

account.

Create a Job to Run the Code

Now that you’ve uploaded the code, you’re ready to automate its execution with a job.

1. In the Databricks workspace, click **Jobs**. Then click **Create Job**.

2. Name the new job **Process Web Log**.

3. For the **Task** option, click **Select Notebook** and select the **ProcessLog** code file you

uploaded previously.

4. For the **Parameters** option, Click **Edit** and add the following key-value pair:

logfile

*Note that these values are case-sensitive.*

IISlog.txt

5. For the **Cluster** option, click **Edit** and configure the cluster for the job as follows:

 **Cluster Type**: New Cluster

 **Databricks Runtime Version**: *Choose the latest available version*

 **Python Version:** 3

 **Driver Type**: Same as worker

 **Worker Type**: *Leave the default type*

 **Workers:** 1

 **Spark Config**: Add a key-value pair for your storage account and key like this:

fs.azure.account.key.***your\_storage\_account***.blob.core.windows.net ***your\_access\_key1\_value***

6. For the **Schedule** option, click **Edit** and note that you can schedule the job to run at regular

intervals. Then cancel the edits to the schedule and leave it set to **None**.

7. Under **Active runs**, click **Run Now** to start the job.

View Job Status and Output

Now that you’ve started the job, you can observe its status and view the output.

1. In the Databricks workspace, click **Jobs**. Note that the **Process Web Log** job is listed with a

green  icon to indicate it is active.

2. Click the **Process Web Log** job. This returns you to the job details page where **Run 1** should

be listed under **Active runs**.

3. Note the **Status** value for **Run 1**. Initially this will be **Pending**. Over time it will change to

**Running**, and then to **Succeeded**. After it has succeeded, **Run 1** will be moved to the

**Completed in the last 60 days** list.

4. Click **Run 1** to view the job output. This includes the code that was executed, and the output

returned by the Spark process that ran the code.

5. In the Databricks workspace, click **Clusters**, and in the **Job Clusters** list, note that a new

cluster was created for the job and is now in a **Terminated** state.

View the Processed Data

Now that you’ve run the job, you can view the processed data.

1. Use your preferred Azure storage tool to view the contents of the **spark** container in your

Azure storage account. This should contain a new folder named **output** that was created by

your job code.

2. In the **output** folder, view the files that have been generated by the job. These include:

 **\_committed\_*nnnn*** (a checkpoint file written when the output was committed)

 **\_started\_*nnnn*** (a checkpoint file written when the job started)

 **\_SUCCESS** (a file indicating the result status of the job)

 **Part-00000-*nnn*** **csv** (the output data produced by the job)

3. Download and view the **Part-00000-*nnnn*** **.csv** file to view the output generated by the job. This file contains the number of web page views for each product.

Challenge 3: 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 zure Event Hubs, Kafka, and others. In this challenge, 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 challenge 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 only **stream\_1.txt** as a block blob

to a new folder named **stream** in root of the **spark** container.

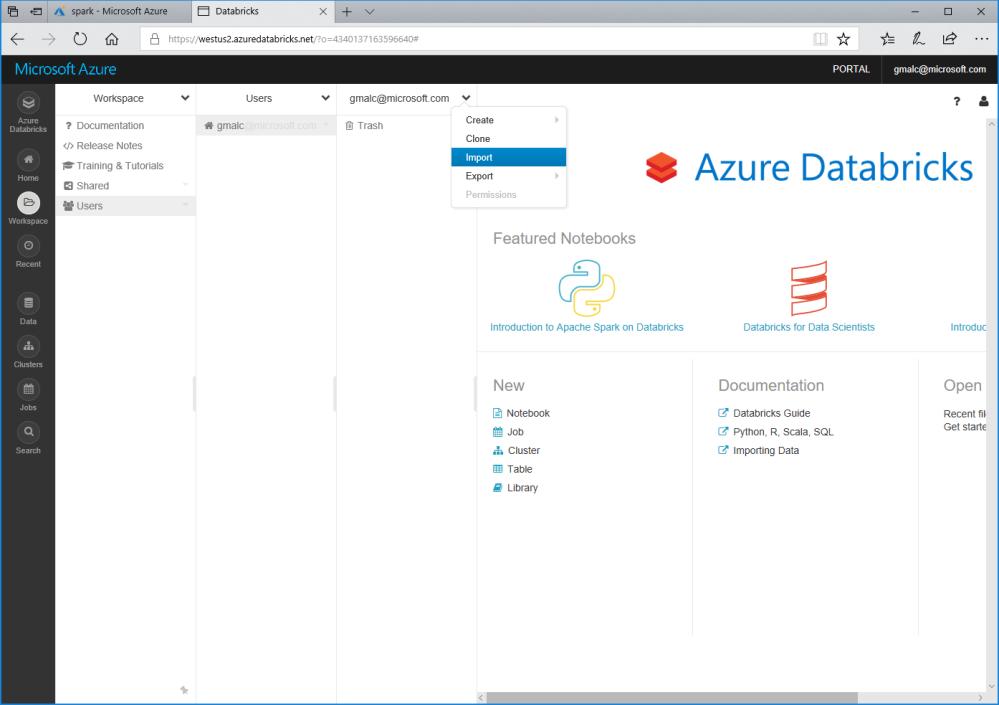
Upload Code to Process the Data Stream

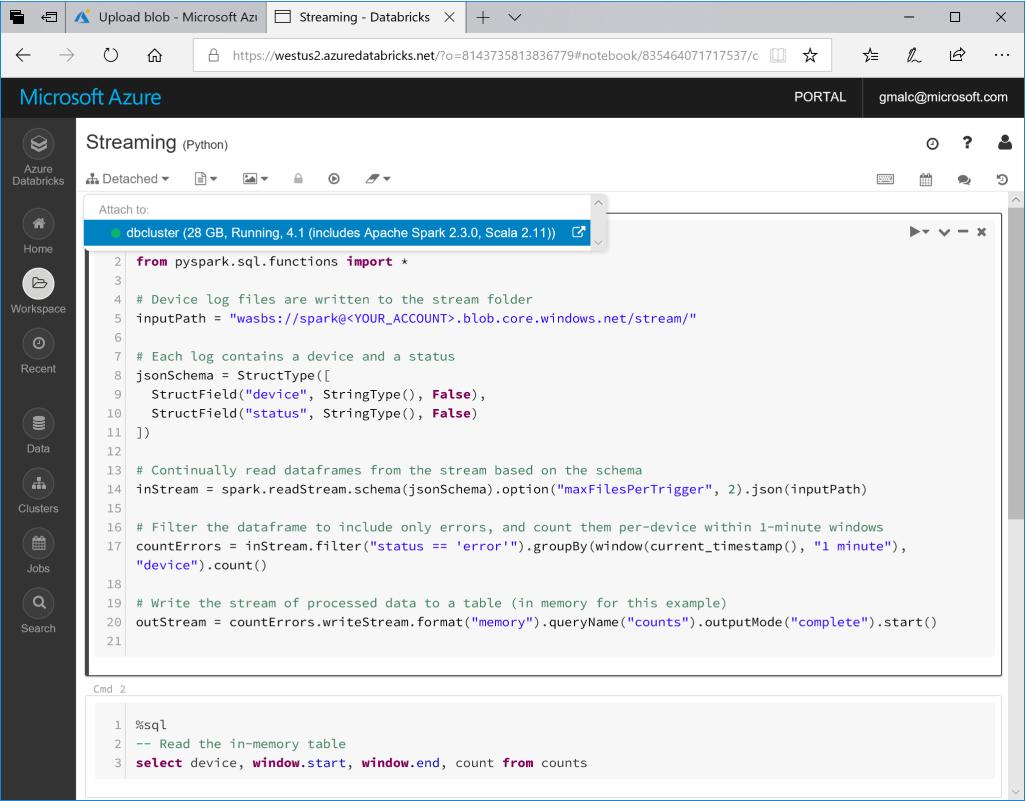
In this challenge, you will use your choice of a Python or Scala script to process the streaming IoT data.

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 challenge 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:

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.

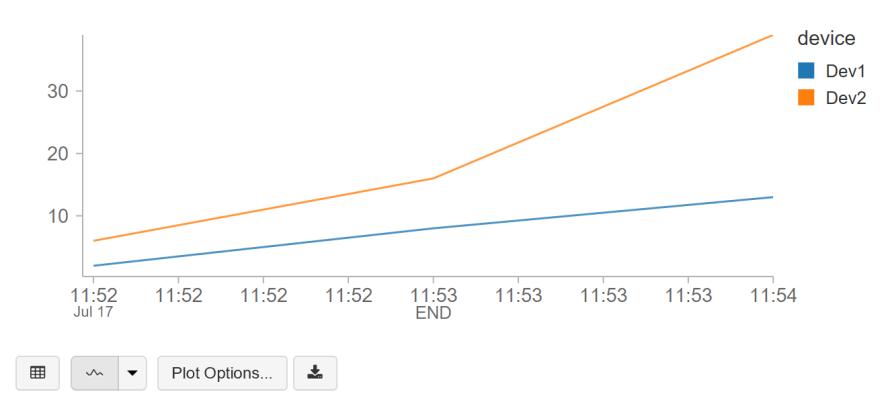
4. Return to the notebook and run the second cell again to verify that the new data added to

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.

Challenge 4: Introduction to Spark for Machine Learning

Background:

Spark includes an API named Spark MLLib (often referred to as Spark ML), which you can use to create machine learning solutions. Machine learning is a technique in which you train a predictive model using a large volume of data so that when new data is submitted to the model it can predict unknown values. The most common types of machine learning are *supervised* learning and *unsupervised* learning.

Machine Learning Terminology:

In a supervised learning scenario, you start with labels that are known & data that also includes *features* (categorical and numeric values that describe characteristics of the entity you’re trying to predict something about). Training the model involves applying a statistical algorithm that *fits* the features to the labels. Because your initial data includes known values for the labels, you can train the model and test its accuracy with these known label values – giving you confidence that the model will work accurately with new data for which the label values aren’t known.

Unsupervised learning is a technique in which there are no known label values, and the model is trained to group (or *cluster*) similar entities together based on their features.

Challenge considerations:

In this challenge, we’ll focus on supervised learning and specifically a type of machine learning called *classification* in which you train a model to identify which category, or *class* an entity belongs to.

Challenge 4.1) Creating and Testing A Machine Learning Model

Upload Source Data to Azure Storage

In this challenge, you will train a classifier to use features of flights that are enroute to an airport, and predict

whether they will be late or on-time. Before you can do this, you must upload the data file containing

the data to your blob storage container where it can be accessed by your cluster. The instructions here

assume you will use Azure Storage Explorer to do this, but you can use any Azure Storage tool you

prefer.

1. In the folder where you extracted the challenge files for this course on your local computer, in the **data**

folder, verify that the **raw-flight-data.csv** files exists. This file contain historic flight data you will

use to train a classification model that predicts whether a flight will be late or not.

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 in this challenge.

4. In the **Upload** drop-down list, click **Upload Files**. Then upload **raw-flight-data.csv** as a block blob

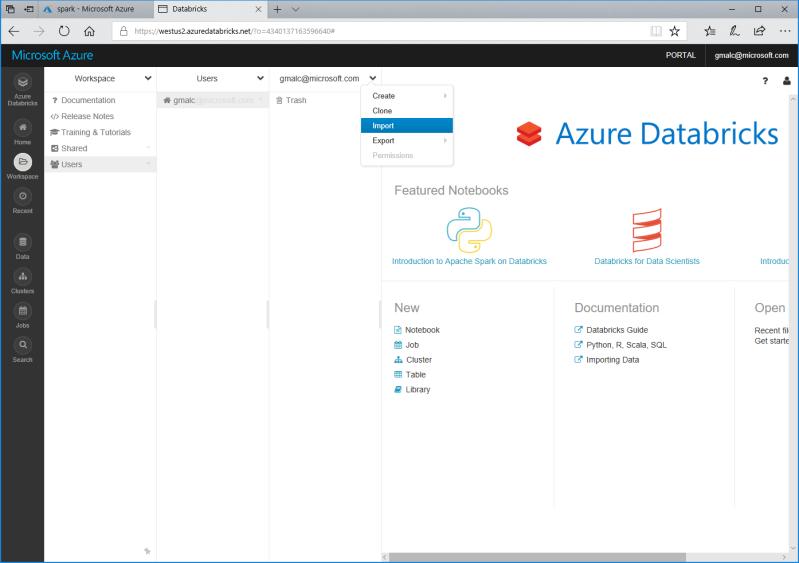
to the **data** folder in root of the **spark** container that you created previously.

Train a Machine Learning Model

In this procedure, you will use your choice of Python or Scala to prepare and explore the flight data,

before training and testing a classification model.

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:

1. Browse to the folder where you extracted the challenge files. Then select either **Machine**

**Learning.ipynb** or **Machine Learning.scala**, depending on your preferred choice of language

(Python or Scala), and upload it.

2. Open the file you uploaded to view the code it contains, read the comments to understand what

it does, and change *<YOUR\_ACCOUNT>* to the name of your Azure Storage account in the first

code cell.

3. Read the notes and run the code cells to explore the data.

Clean Up

**Note**: This is the final challenge in this course. If you have finished exploring Spark, 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.

## Working together

* Will your team work together or individually?
* Git for Windows – allows for working with Git repos locally on your machine as well as in VSTS. <https://gitscm.com/downloads>
* Determine Azure Subscription you will use for deployment

## Resources

There are several folders in the Team site, including resources folders for each challenge.

# Challenges (attendee view) : ***Got Data? Use Spark in Azure Databricks***

## Challenge 1: Intro to Spark

This challenge involves loading textual data in Azure blob storage (for this exercise, we will be using President Kennedy’s inaugural speech as our content source) and will use Spark - either python or scala (your choice) - to filter and transform the dataset according to the challenge summary.

**Challenge Summary**

Your team is preparing a proof of concept, and your group has been tasked to suggest answers to the following questions:

1. Can you determine the intent of President Kennedy’s speech <hint: look for words like *freedom* & create filtered views for those keywords to answer the question>

2. Are there any additional Spark concepts that you would use to help bring value to your analysis?

3. What are some potential ways you could save time during your analysis <hint: less lines of code following one API approach vs. another means faster time to value>

4. Are there any notable differences when analyzing Unstructured vs. Structured datasets?

Document your suggestions on the teams site or other whiteboards/flipcharts provided. You will present your results.

Business Case Ia – Unstructured Analysis using President Kennedy’s Speech – <need to add link here>

* Setup assistance & Introduction to Azure Databricks concept <See 1\_Setup.md in the Team site > - includes setting up Azure Blob Storage for data files that will be consumed during challenges
* Challenge 1.1: Setting up Azure Databricks: Create a cluster
* Challenge 1.2: Setting up Azure Databricks: Create a workspace & your first notebook
* Challenge 1.3: Use RDD API (python or scala) to transform unstructured data file from Azure Blob Storage

Business Case Ib – Structured Analysis of Traffic Accidents – <need to add link here>

* Challenge 1.4: Setting up Azure Databricks: Create new notebook & label it **Dataframes.ipynb**
* Challenge 1.5: Use Dataframes API (python or scale) (Spark 2.0) to analyze new STRUCTURED datasets (related to traffic accidents) : Read data in from Azure Blob storage
* Challenge 1.6: Create dataframes to analyze traffic patterns & vehicle densities by location;
* Challenge 1.7: Create top and bottom views of accident data to better under emerging patterns <if they exist>
* Challenge 1.8: Transform existing dataframes into Temporary Tables or Permanent Tables if your query results are deemed meaningful by your table using Spark SQL API

Follow up Activities:

* Discuss approaches and analysis with Table
* Load your Q&A and Feedback on your Teams site

# ***Got Data? Use Spark in Azure Databricks***

## Challenge 2: Running a Spark Job

What You’ll Need

To complete the challenges, 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 challenge files for this course <only if you get stuck – proctors have access to this content>

**Note**: To set up the required environment for the challenge, follow the instructions in the [**Setup**](https://microsoft-my.sharepoint.com/personal/laedell_microsoft_com/Documents/Hackathons/ML/Databricks/databricks_intro/databricks-introFinal/Setup%20Guide.docx)document for

this course. Specifically, you must have signed up for an Azure subscription.

Challenge Background: Spark jobs enable you to run data processing code on-demand or at scheduled intervals. This enables you to build data processing solutions for unattended execution.

Business Challenge: your business has asked you to analyze voluminous web server logs

Questions:

1. What types of analyses would be good when working with website log data ? Remember, you want to provide value to the business quickly.
2. If you look at the ProcessLog.py files, what Spark modules are being used and why do you think they were chosen? Would you have chosen alternate modules and why?
3. What else would you add to the script to enhance what it is already doing using what you learned in Challenge 1?
4. Can you explain the purpose of the different files that were created by the Spark job process in your blob storage?
5. What is the file name(s) created that contains the Processed Results? Does it contain the # of webpage views for each product that you were expecting?

To get you started, the 1st thing you will need to do is to Import your ProcessLog.py or .scala script into your Databricks Workspace. How? Ask your table mates.

\*\* remember: you can use scala or python; you can use Databricks notebooks or native scripts (.py, for example) to solve the Challenge \*\*

\*\*\*Lastly. If you get stuck, proctors can help you without providing the answer.

**Challenge 2 Outline:**

Challenge 2.1) Access Databricks Workspace from Challenge 1 or Provision a New Databricks Workspace if you didn’t complete Challenge 1

Challenge 2.2) Access existing Storage Account & Create a new Container for Challenge 2 (if you didn’t complete Challenge 1, create a new Storage Account / Container)

**helpful tips:**

\*\*Upload the log source files - These logs have been made available as part of the Teams site under Files <IISLog.txt>

Challenge 2.3) Create Spark Job in Azure Databricks

**helpful tips:**

\*\* you will need to import the ProcessLog.py or ProcessLog.scala file from Teams to your Databricks Workspace (file found in Teams files)

\*\*you will need to edit the scripts with your storage account details – also, add other transformers or use different modules if you want to test out your knowledge or have a different idea for how to approach the challenge – points rewarded for teamwork and for creative thinking!!

**Challenge considerations:**

1. Access your Databricks workspace to create Spark job – Spark jobs can be used to automate execution of your scripts. Spark jobs that are created can be checked into a DevOps CICD pipeline along with artifacts created in your Databricks workspace (eg. Notebooks and other script files).

2. For the **Task** option, click **Select Notebook** and choose the **ProcessLog** code file you

uploaded previously (either .py or .scala)

3. For the **Parameters** option, enter the following key-value pair **<note: values are case sensitive>**

logfile

IISlog.txt

4. **Cluster** settings - , click **Create New (this is your jobs cluster):**

• Cluster Type: New Cluster

• Databricks Runtime Version: *Choose the latest available version*

• Python Version: 3

• Driver Type: Same as worker (leave default)

• Workers: 1

* **Important setting to append:**

• **Spark Config**: Add a key-value pair for your storage account and key like this:

fs.azure.account.key.***your\_storage\_account***.blob.core.windows.net ***your\_access\_key1\_value***

5. For the **Schedule** option, choose a schedule if you want OR click **Run Now** to start the job.

6. View job status as it moves from **Pending to Running to Succeeded.**  Notice how Active Runs changes as status change

**Challenge 3 Outline:**

**Challenge 4 Outline:**

## Working together

* Will your team work together or individually?
* Git for Windows – allows for working with Git repos locally on your machine as well as in VSTS. <https://gitscm.com/downloads>
* Determine Azure Subscription you will use for deployment

## Resources

There are several folders in the Team site, including resources folders for each challenge.