**DNN Workshop**

**Introduction**

This delivery of the DNN workshop will use an Azure service called *Databricks*. *Databricks* is a cloud managed service that provides a spark infrastructure for building machine-learning and deep-learning data-pipelines. Think about it as an interactive workspace built around managed spark clusters that exposes R, Scala, Python, and SQL interfaces. Databricks notebooks provides the interactive UI and if you are accustomed to working with Jupyter or Zeppelin notebooks, it will feel familiar. It is the eponym product by a San Francisco based company called *Databricks* and you can learn more about it [here](https://databricks.com/product/faq).

A [Spark glossary](#Glossary) is provided towards the end of this document.

**Objectives**

In this hands-on lab, you will learn how to setup a *Databricks* workspace, create a managed spark cluster in a few minutes, and start working with *Databricks* notebooks. For this lab we will be using Python and azure based storage.

**Prerequisites**

The following are required to complete this hands-on lab:

* An active Microsoft Azure subscription. If you don't have one, [sign up for a free trial](http://aka.ms/WATK-FreeTrial).
* Sufficient CPU or Graphical Processing Units (GPUs) to create a Databricks cluster. Check to see if your [subscription has a quota](https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits) for CPUs or GPUs (NC6 GPUs). If you need to increase the quota, [open a Support Request from your azure portal](https://docs.microsoft.com/en-us/azure/azure-supportability/resource-manager-core-quotas-request).
* Optional: GPUs are often used for compute intensive workloads like deep learning. During the lab you have the option of provisioning a Databricks cluster with a GPU (an NC6 is the minimum GPU configuration). Be aware that today these GPU-enabled Databricks clusters are in Beta.

**Exercises**

This hands-on lab includes the following exercises:

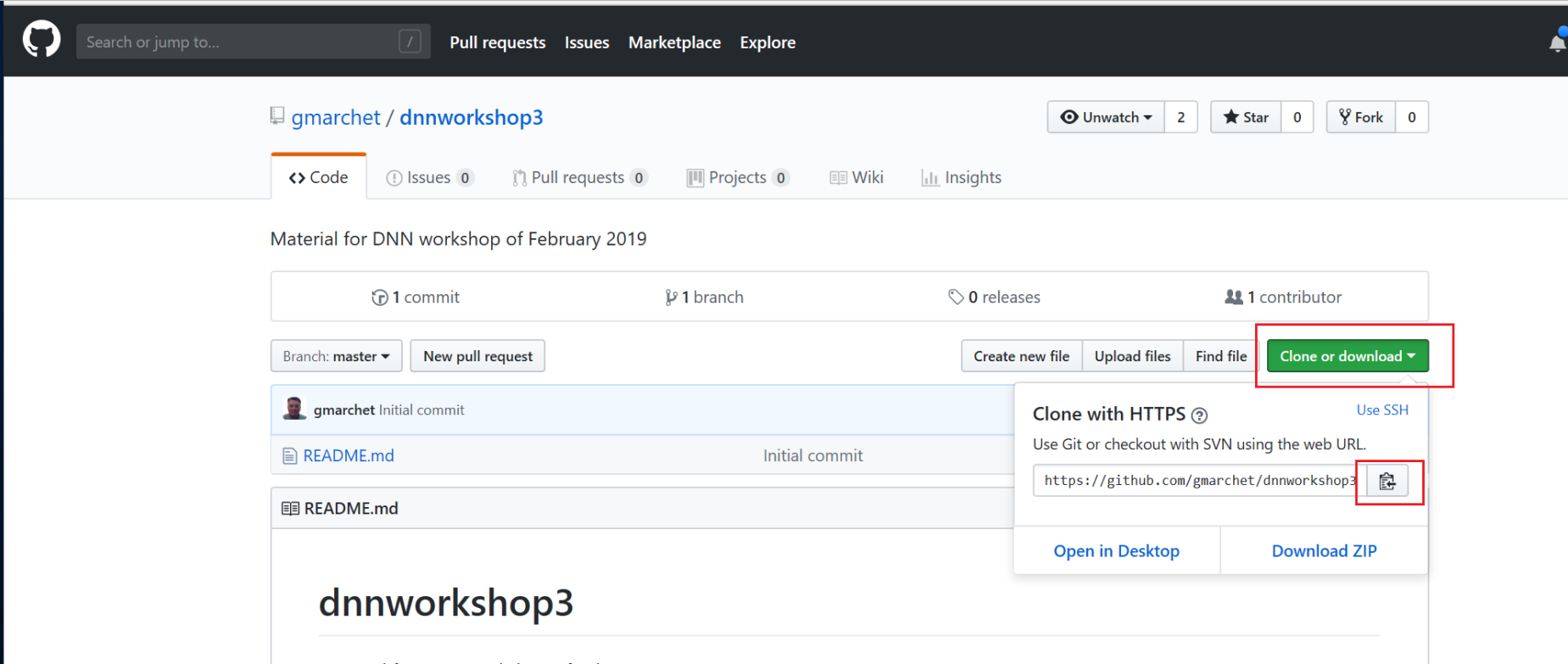
* [Exercise 1: Download the DNN workshop github repo to your local machine](#Excersize1)
* [Exercise 2: Create an Azure Databricks workspace](#Excersize2)
* [Exercise 3: Create a Databricks Machine Learning clus](#Excersize3)ter
* [Exercise 4: Install libraries to the cluster](#Excersize4)
* [Exercise 5: Import a python notebook into the workspace](#Excersize5)
* [Exercise 6 (optional): Create an azure storage account and container, upload a data file, and mount the azure storage blob to the Databricks cluster](#Exersize6)

Estimated time to complete this lab: **60-90 mins.**

**Exercise 1: Download the github repo to your local machine**

Content for this workshop is uploaded into a github repo here: <https://github.com/gmarchet/dnnworkshop3>. In this exercise you will download the content from the repo onto your local laptop. The content includes the notebooks we will use in the workshop labs.

1. Navigate to the git repository and click on ‘*Clone or download*’ and select the button to copy the URL:



*GitHub repo for DNN Workshop*

1. Bring up a command window and type in

git clone <paste in the URL from github>

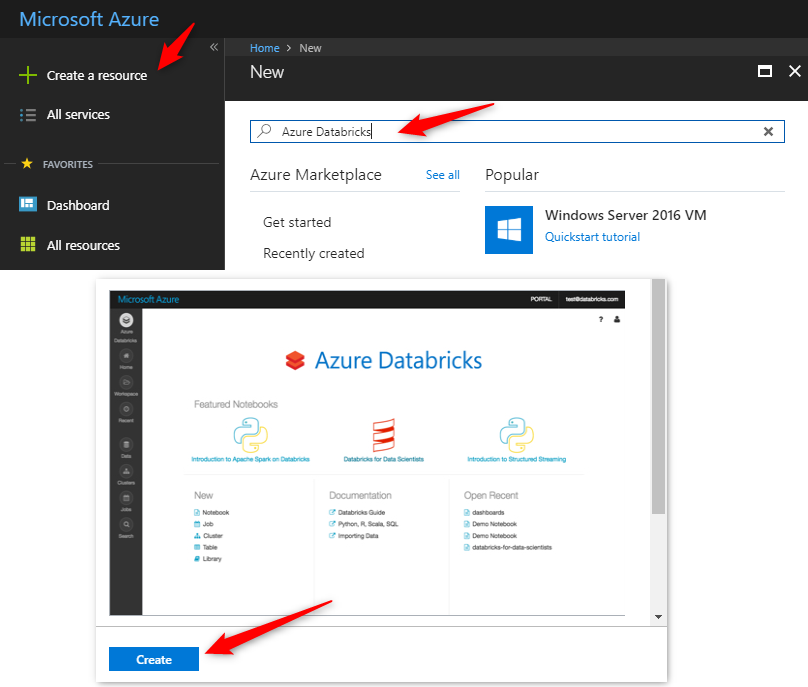
1. This will copy the *dnn-workshop3* repo to a local folder

With this download to your local machine, you have all the course content and are ready to get started.

**Exercise 2: Create a Databricks Workspace**

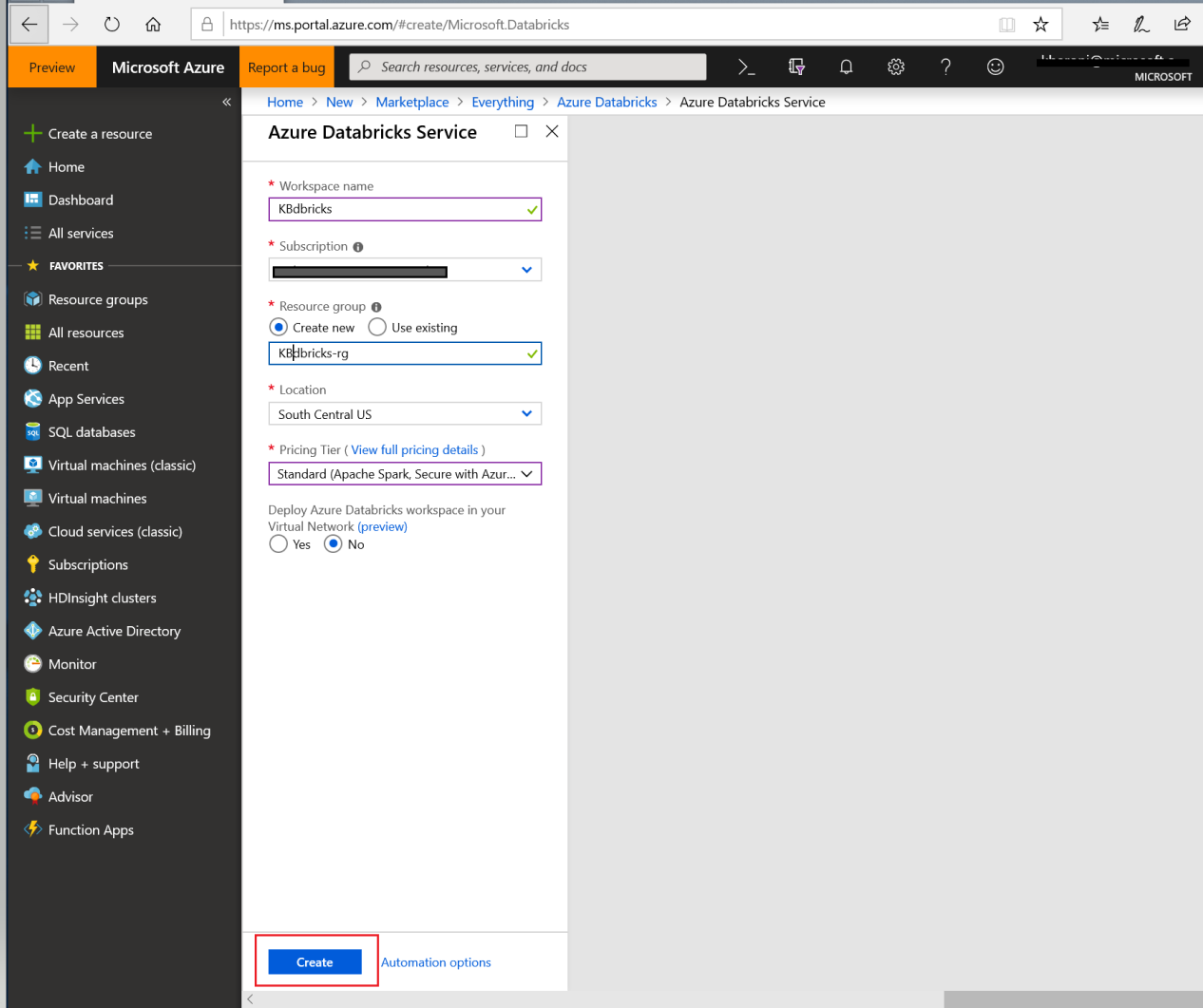
In this exercise, you will create a *Databricks* workspace. The workspace stores all of your notebooks, libraries, and dashboards.

1. Open the [Azure Portal](https://portal.azure.com) in your browser. If asked to log in, do so using your account.
2. Click **+ Create a resource** in the menu on the left side of the portal, and then type "Azure Databricks” (without quotation marks) into the search box. Select **Azure Databricks** from the results list, take a moment to review the information about the service, and click **Create**.



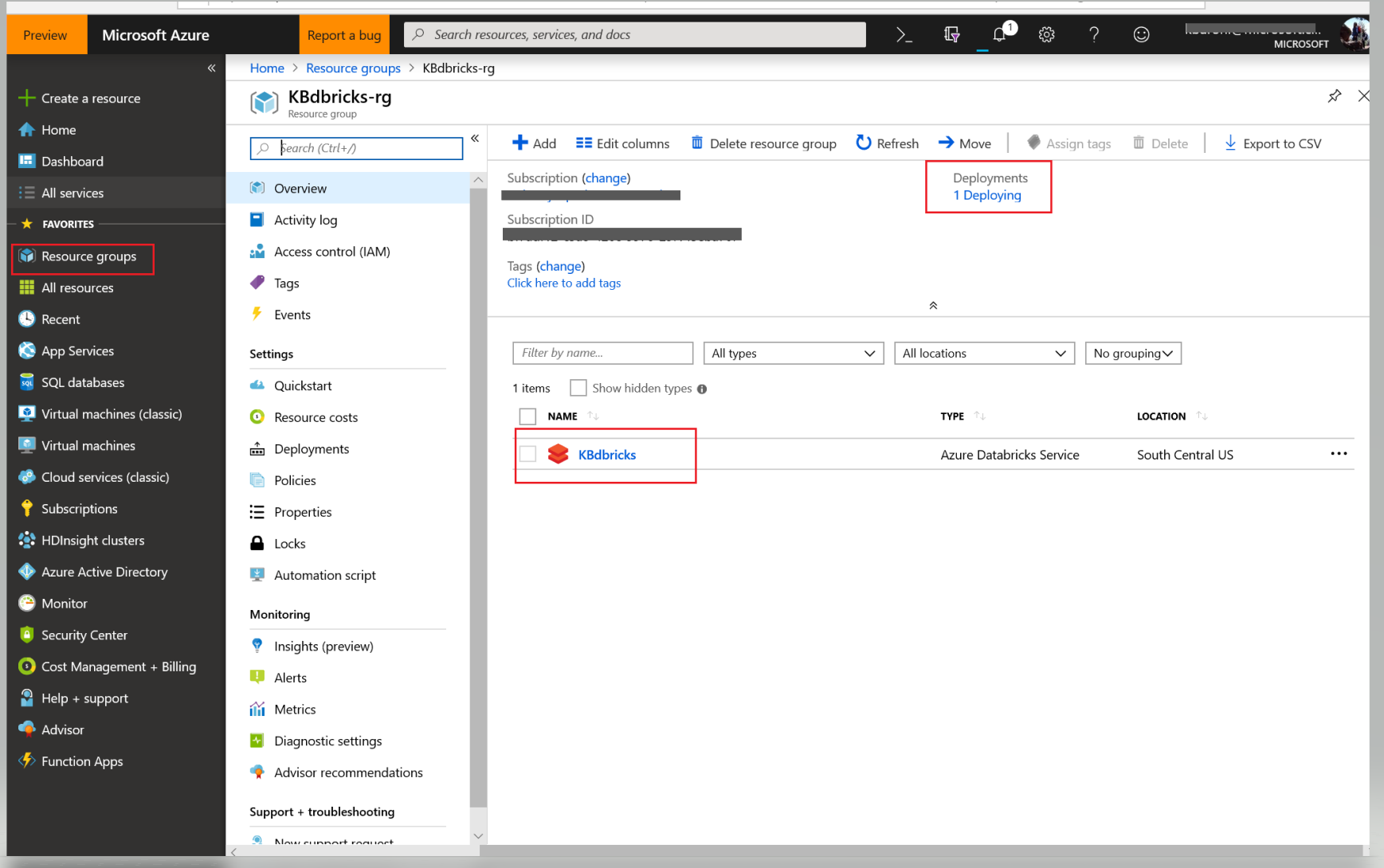
*Create a Databricks workspace*

1. Enter a name for the virtual machine, select the subscription to use. Select **Create new** under **Resource group** and enter a resource-group name such as "databricks-rg." Select the **Location** nearest you, the *Standard* pricing tier, and then click **Create**.



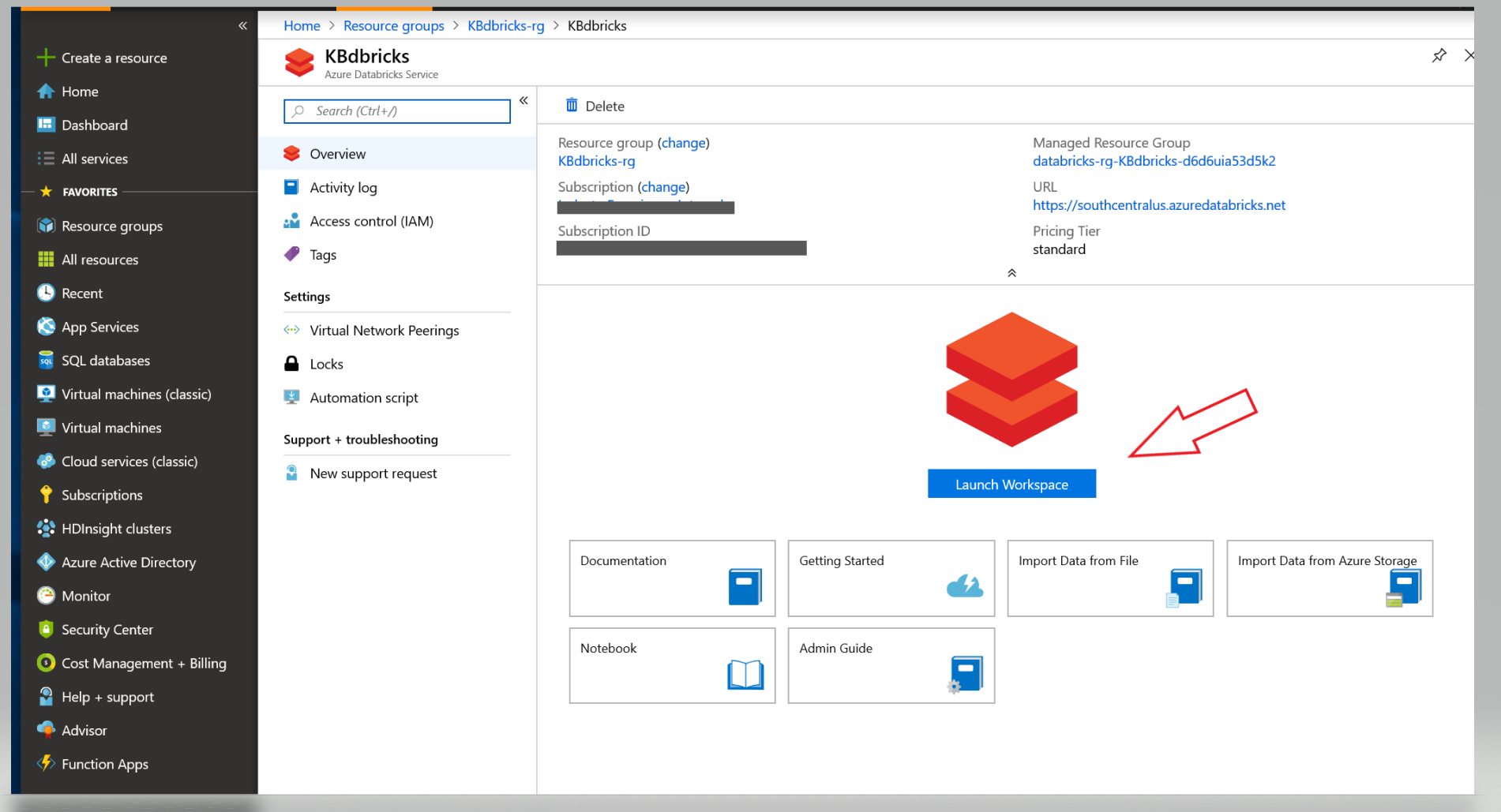
*Enter basic workspace settings*

1. Click **Resource groups** in the menu on the left side of the portal. Then click the resource group whose name you specified in the step above.



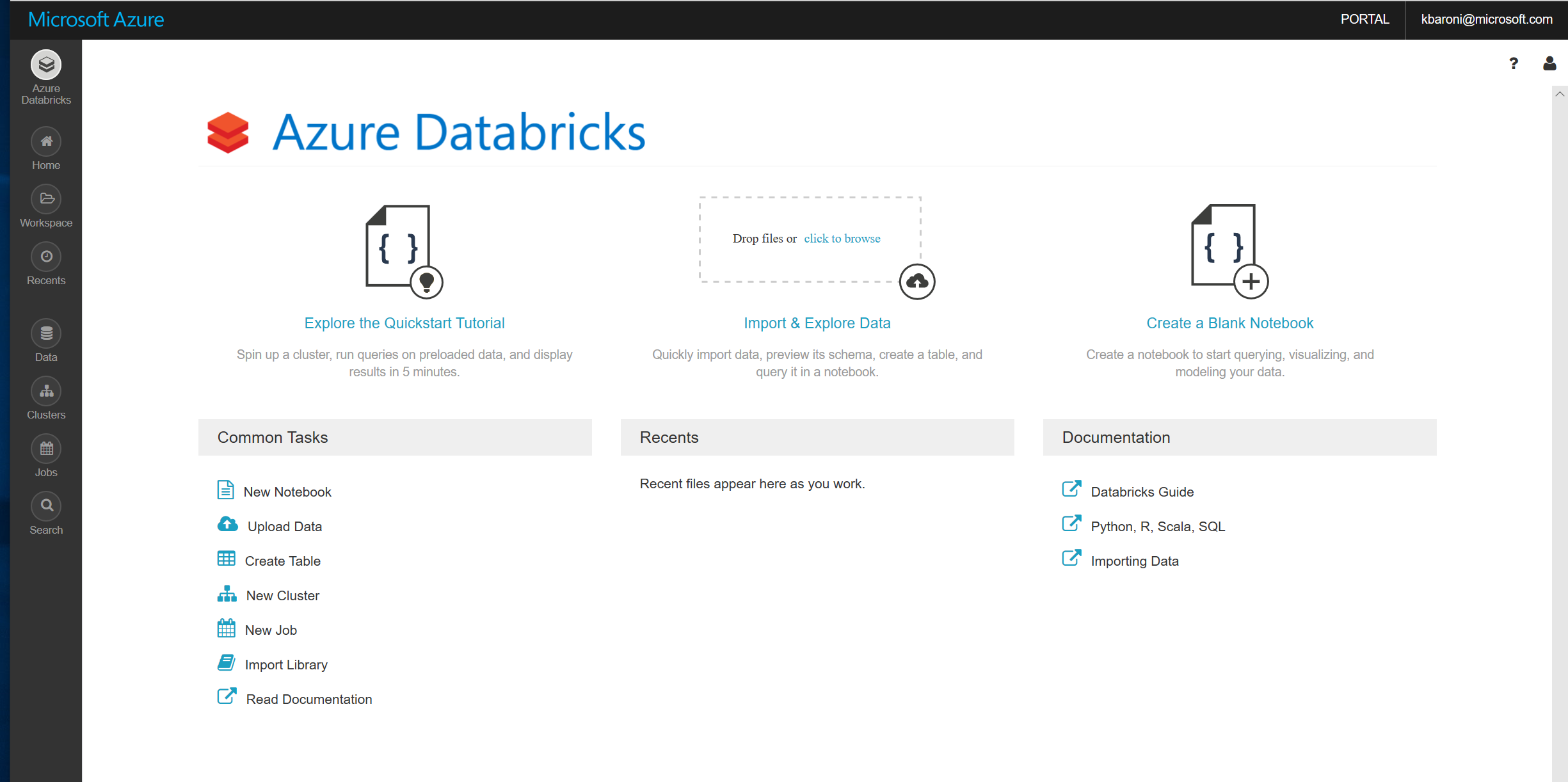
*Open workspace resource group and monitor deployment*

1. Wait until "Deploying" changes to "Succeeded" indicating that deployment has completed. Deployment typically takes less than a minute. Once deployment has succeeded, select the resource and click on ‘Launch Workspace’:



*Launch the Databricks workspace*

You will be signed into *Azure Databricks* service and when the screen below appears, ready to get to work:



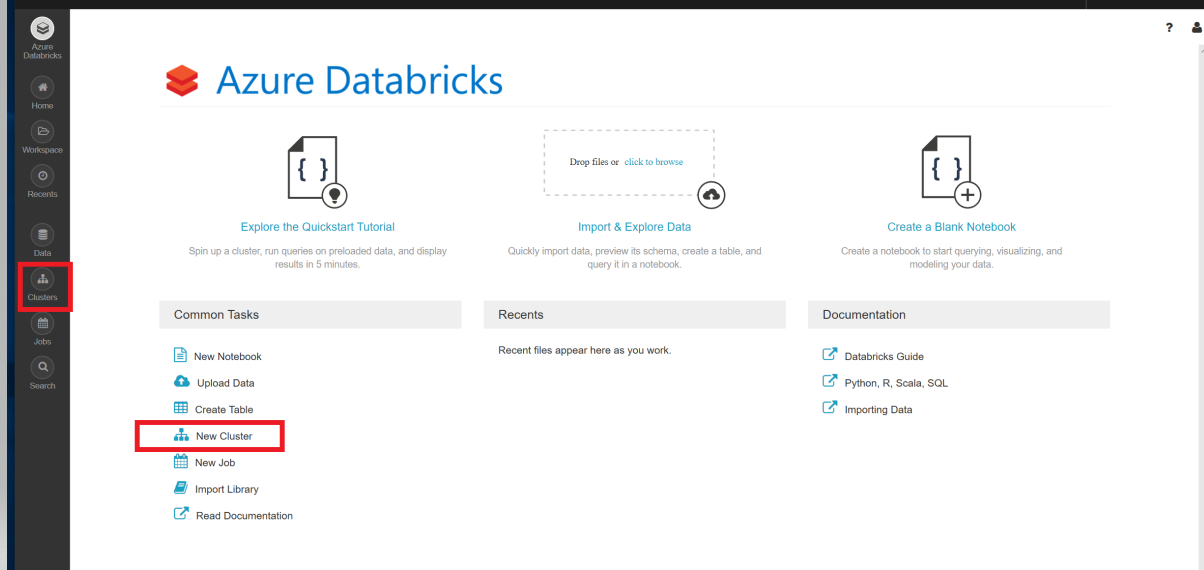
*Databricks workspace*

**Exercise 3: Create a *Databricks Spark* cluster (ML Cluster is optional)**

Spark clusters are the foundation of Azure Databricks. In this exercise, you will create a Databricks cluster.

Optionally you can create a special version of a *Databricks* cluster **in Beta** called *Databricks Runtime ML*. This version provides an environment already configured with popular AI libraries like TensorFlow, PyTorch, Keras, and XGBoost. It also supports a distributed training environment using [Horovod.](https://pypi.org/project/horovod/) The cluster you provision will be a GPU-enabled cluster optimized for deep learning workloads. More information about this runtime is available here: [Overview of Databricks Runtime for Machine Learning](https://docs.azuredatabricks.net/user-guide/clusters/mlruntime.html#create-a-cluster-using-databricks-runtime-ml).

1. To create the cluster, Select *New Cluster* from **Common Tasks** OR click on the *Cluster icon* in the left side bar:

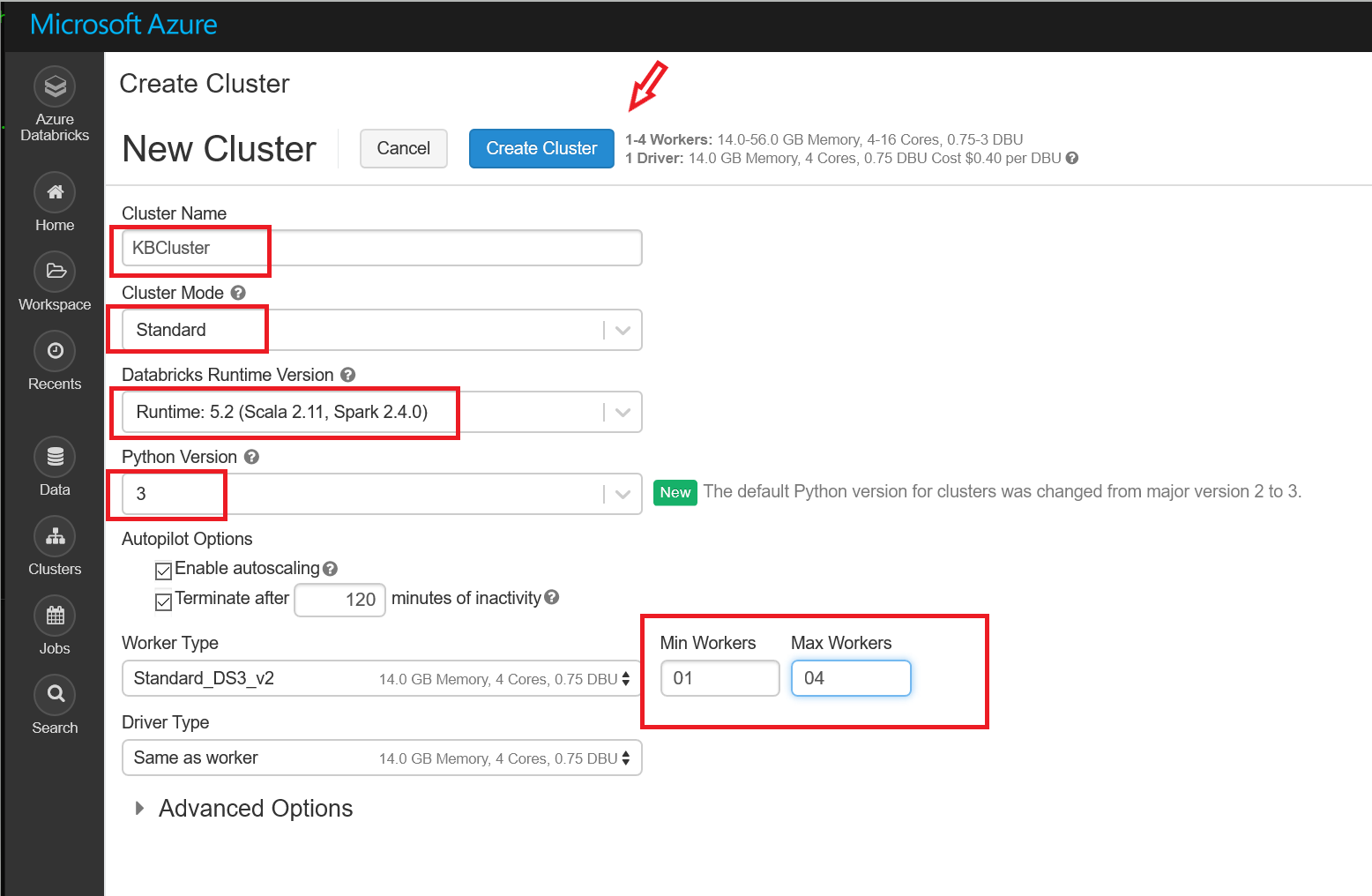


*Select New Cluster*

1. After designating a name for your cluster, configure it using the specifications below . (Optionally for the GPU-enabled cluster) For the workshop, we are going to configure the cluster with a minimum of 1 worker node Keep in mind you can re-configure cluster size at a later time.

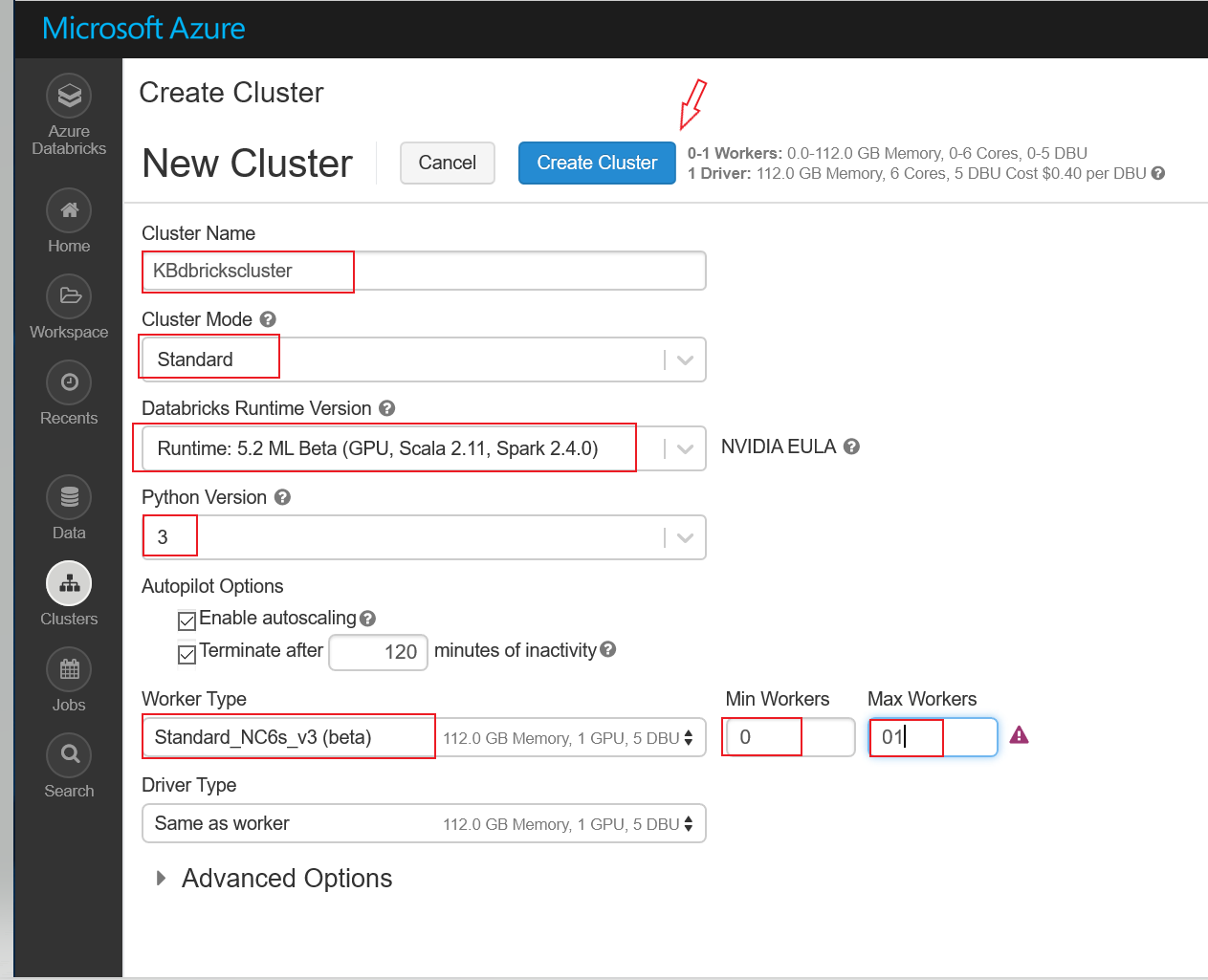
Also note the default setting to shut down the cluster after 2 hours. This is a convenient fail-safe but keep in mind, recommended practice is to shut down your cluster when you are not using it and not rely on auto-pilot default settings.

When done with the configuration settings, click Create Cluster. It can take several minutes for the cluster to provision



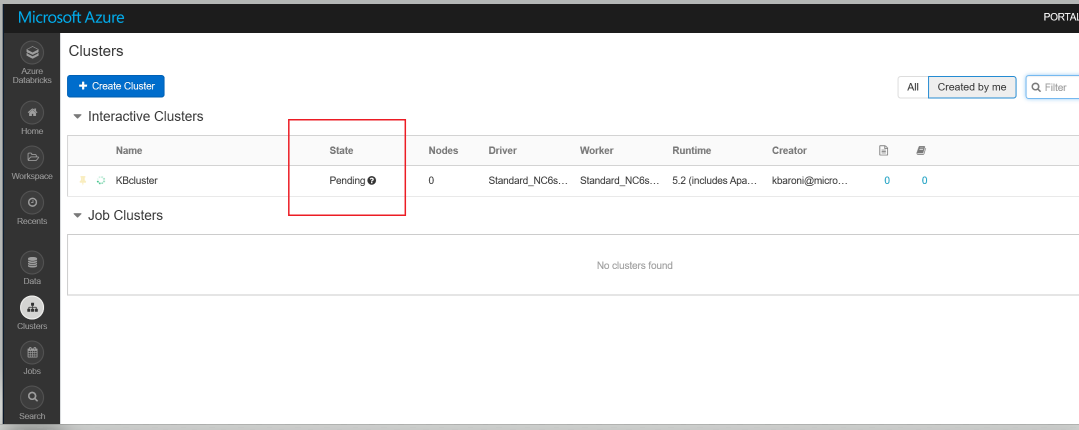
*Configure a cluster for CPU*

For a GPU-enabled Databricks cluster:



*Configure a cluster for GPU*

1. Cluster status will indicate *Pending* while the service is being provisioned. Once complete, status will register as *Running*.

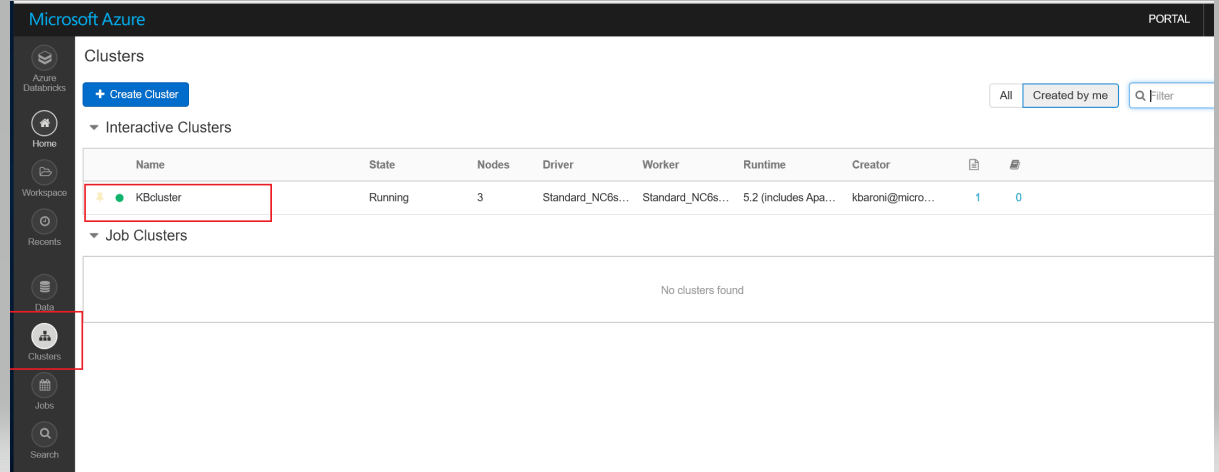


*Cluster service is provisioning*

**Exercise 4: Install libraries**

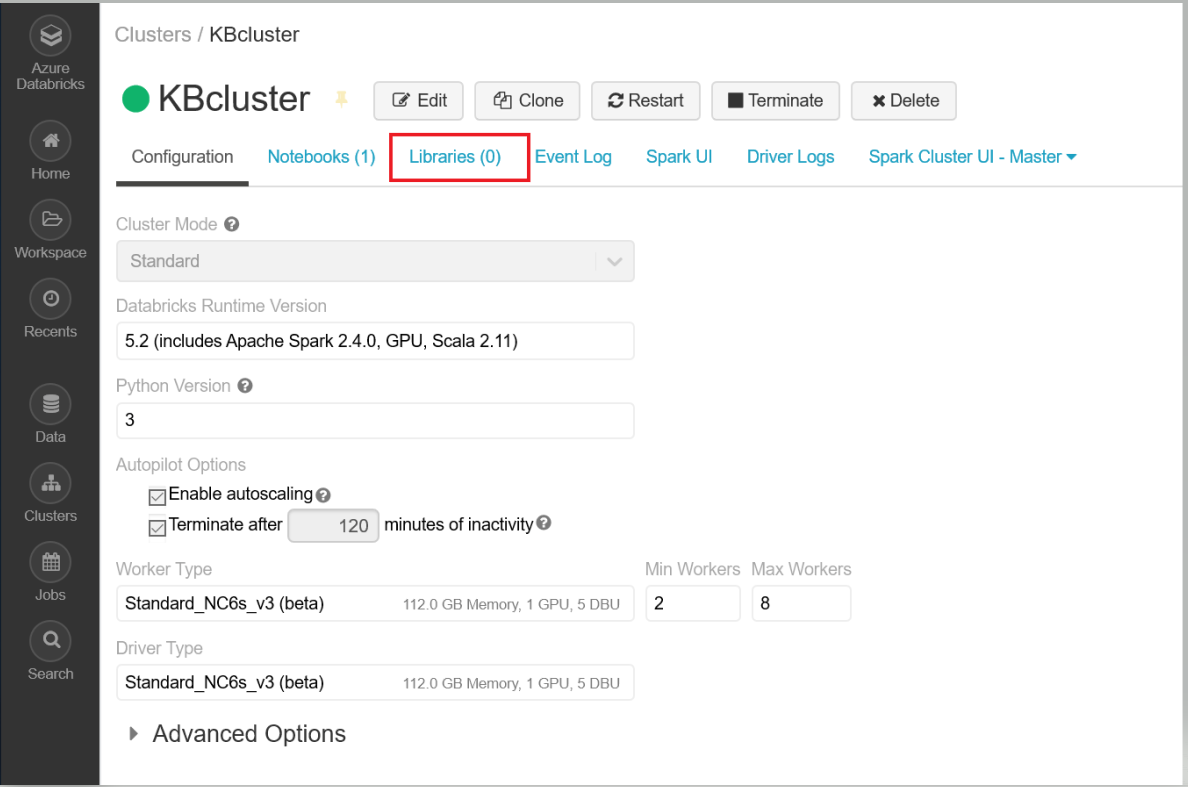
In this exercise, you will install a python libraries on the cluster. Once a library is installed, it is available to all users of the cluster. Note: If you are setting up a CPU-enabled cluster, you need to install these libraries on your cluster. If you have provisioned an ML/GPU-enabled Databricks cluster, these libraries will already be available.

1. Select the clusters icon, bring up the list of clusters, and select the cluster you are using for this lab:



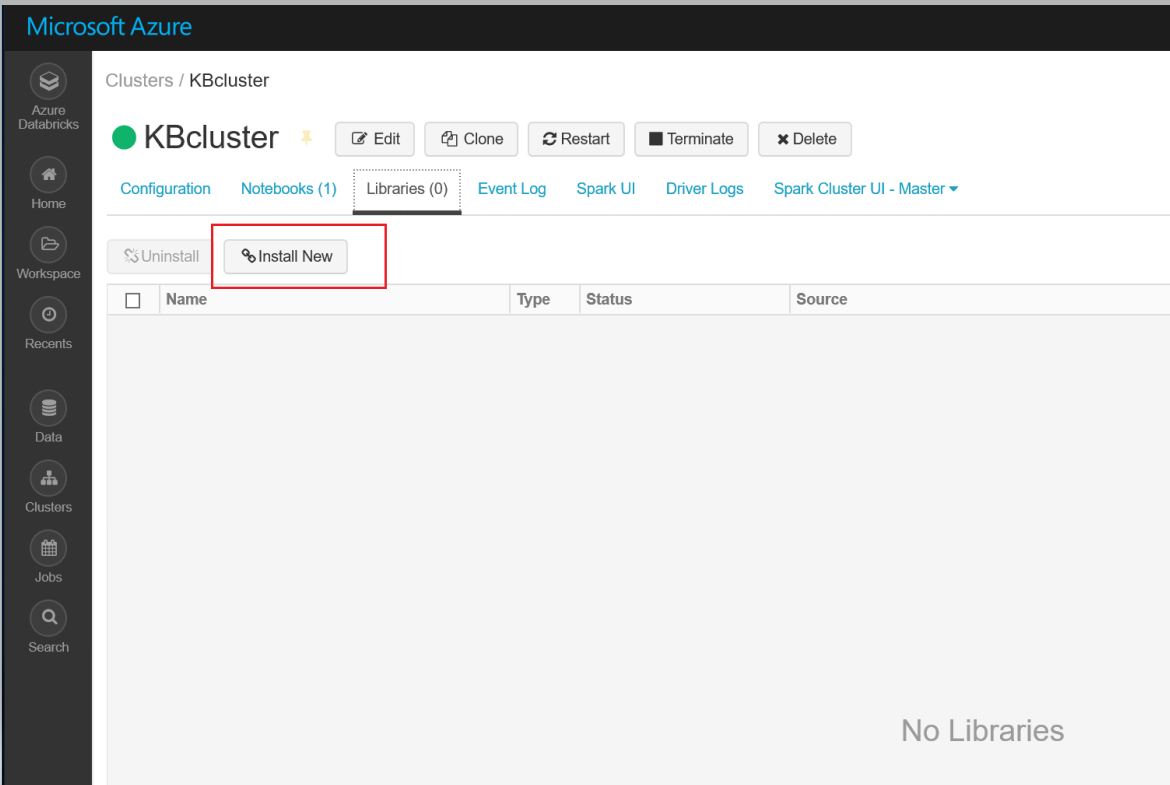
*Select the target cluster*

1. Select **Libraries**



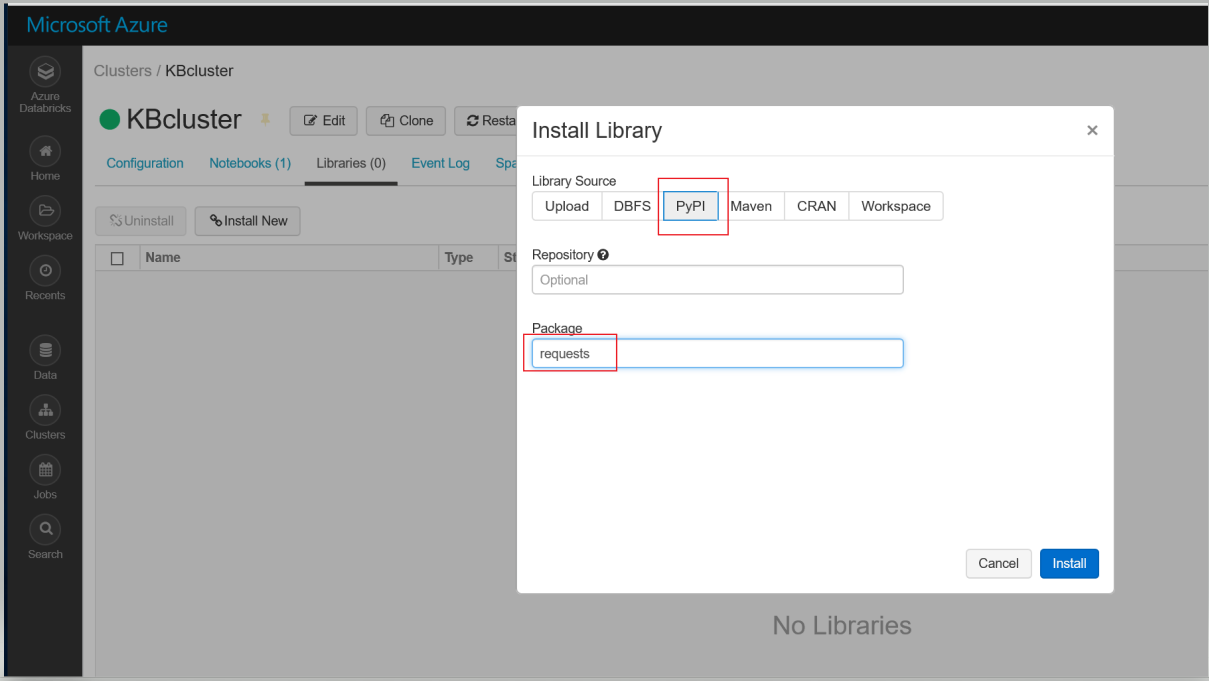
*Click on the ‘Libraries’ tab*

1. And **Install New**



*Install a new library*

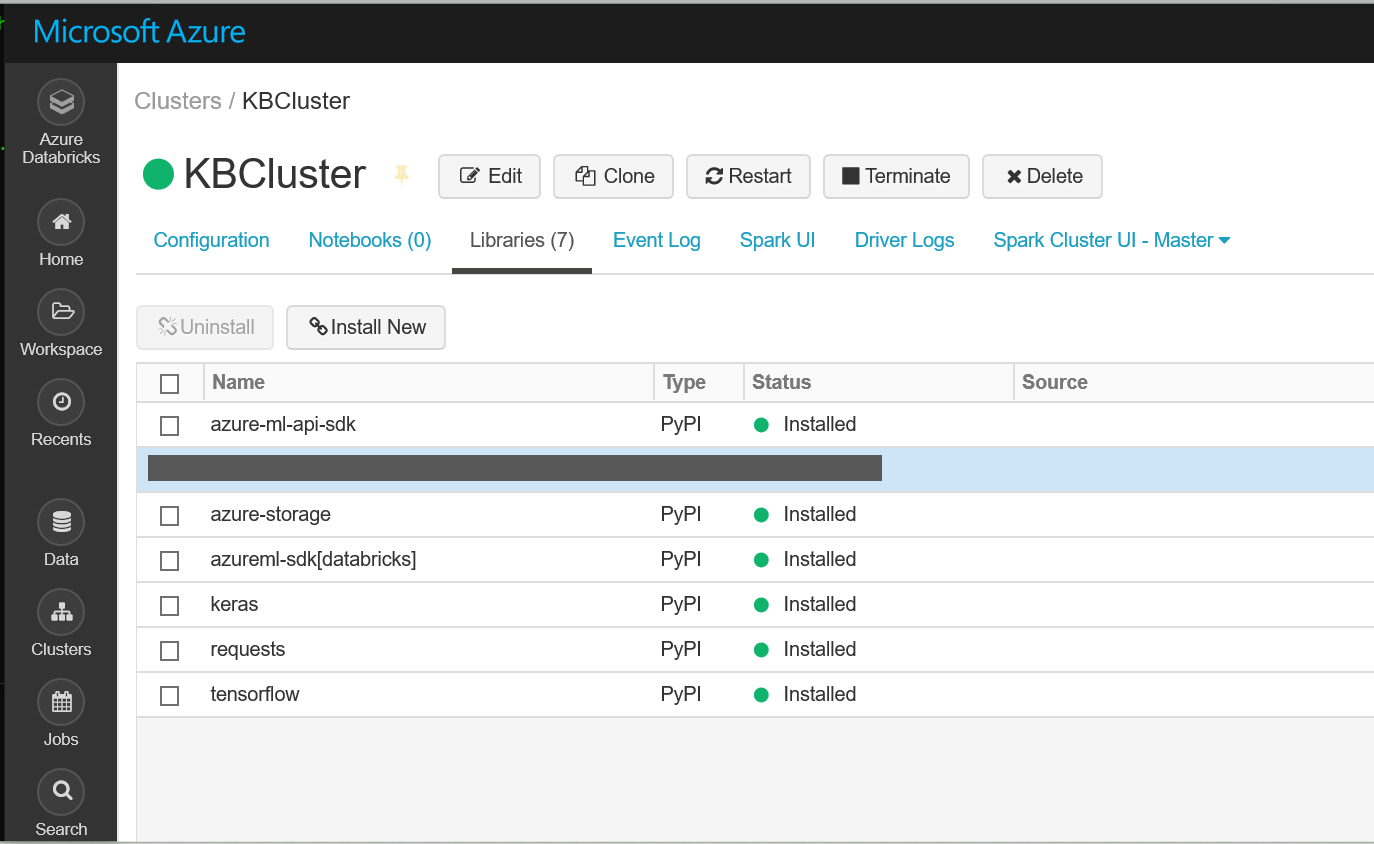
1. Select *PyPl* as the library source, the name of the library package (for instance *requests*) and select **Install**



*Identify the library source and package to import*

***Congratulations!*** You have installed your first library on your cluster. Repeat **Step 4** in this exercise to add these additional libraries:

**azure-ml-api-sdk, azure-storage, azureml-sdk[databricks], keras, tensorflow.**

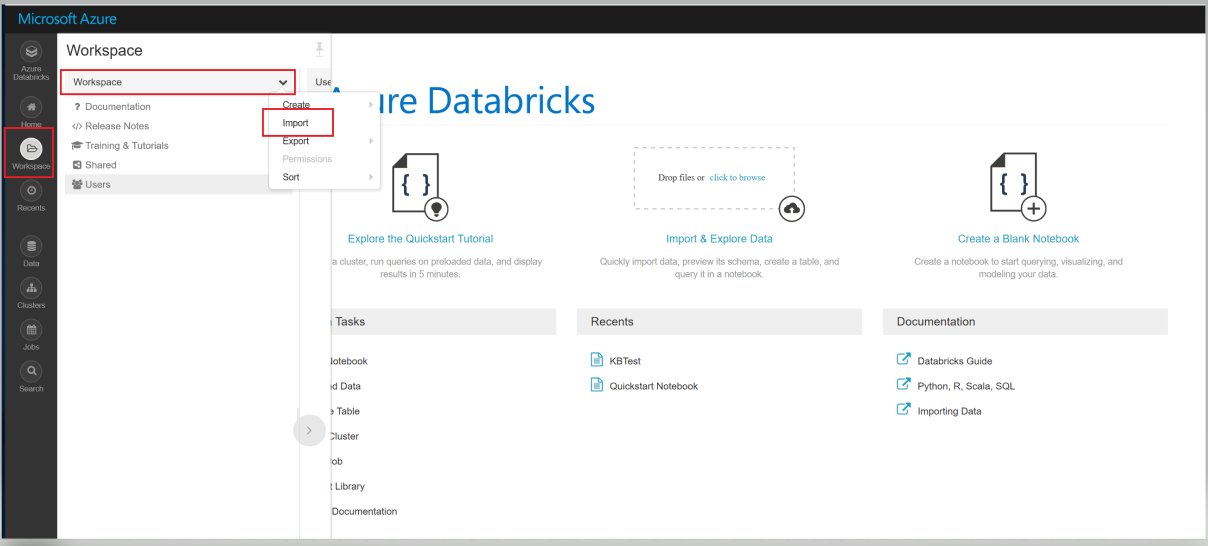


*Install additional libraries needed for the labs*

**Exercise 5: Import a Python notebook**

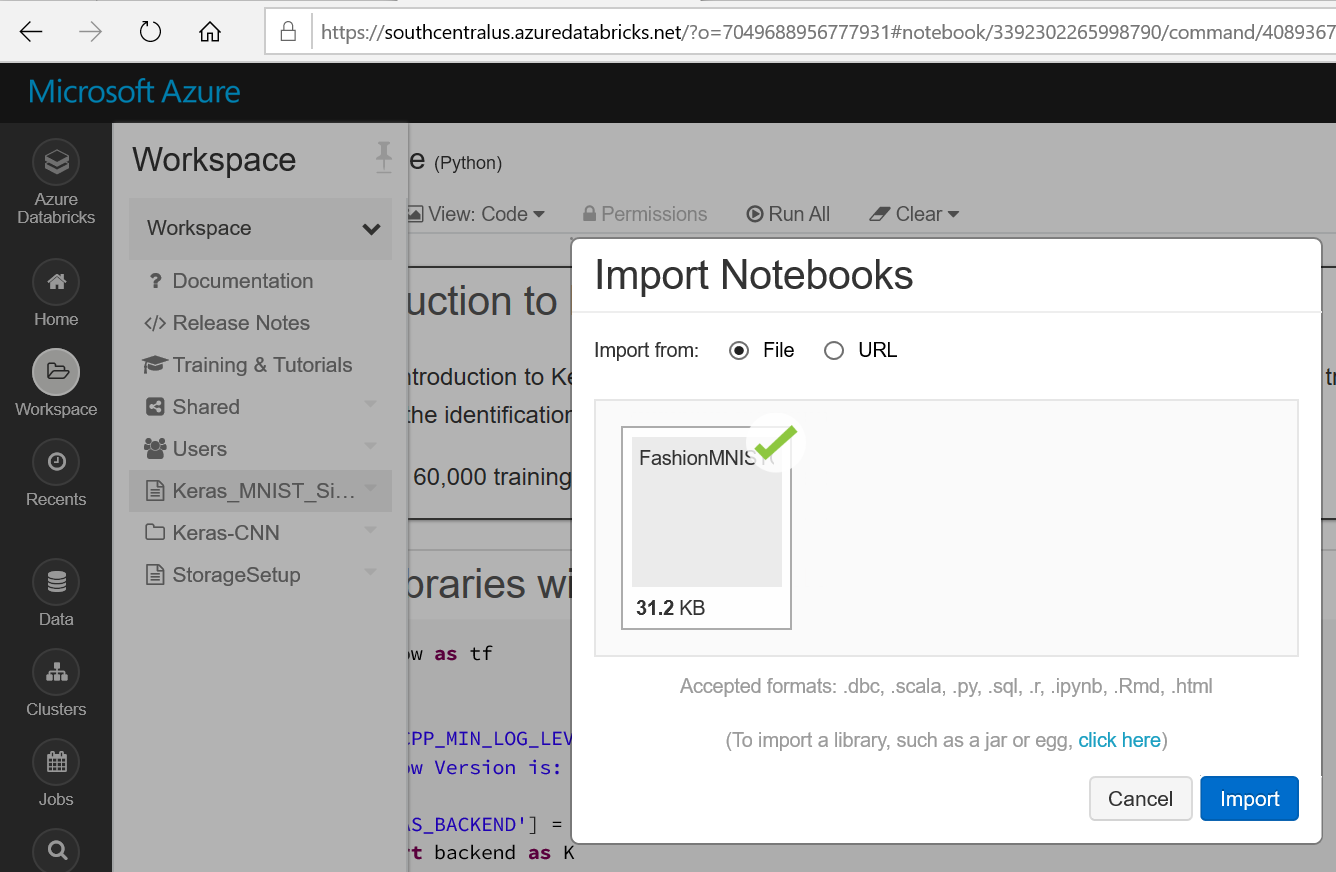
During this exercise we will import a python file (.ipynb) from the downloaded github repo into the Databricks workspace.

1. Click on the Workspace button to navigate back to the workspace area. To the right side of Workspace column, click the down arrow and select *Import* (**note:** you may have to refresh the browser to get the import option to display)



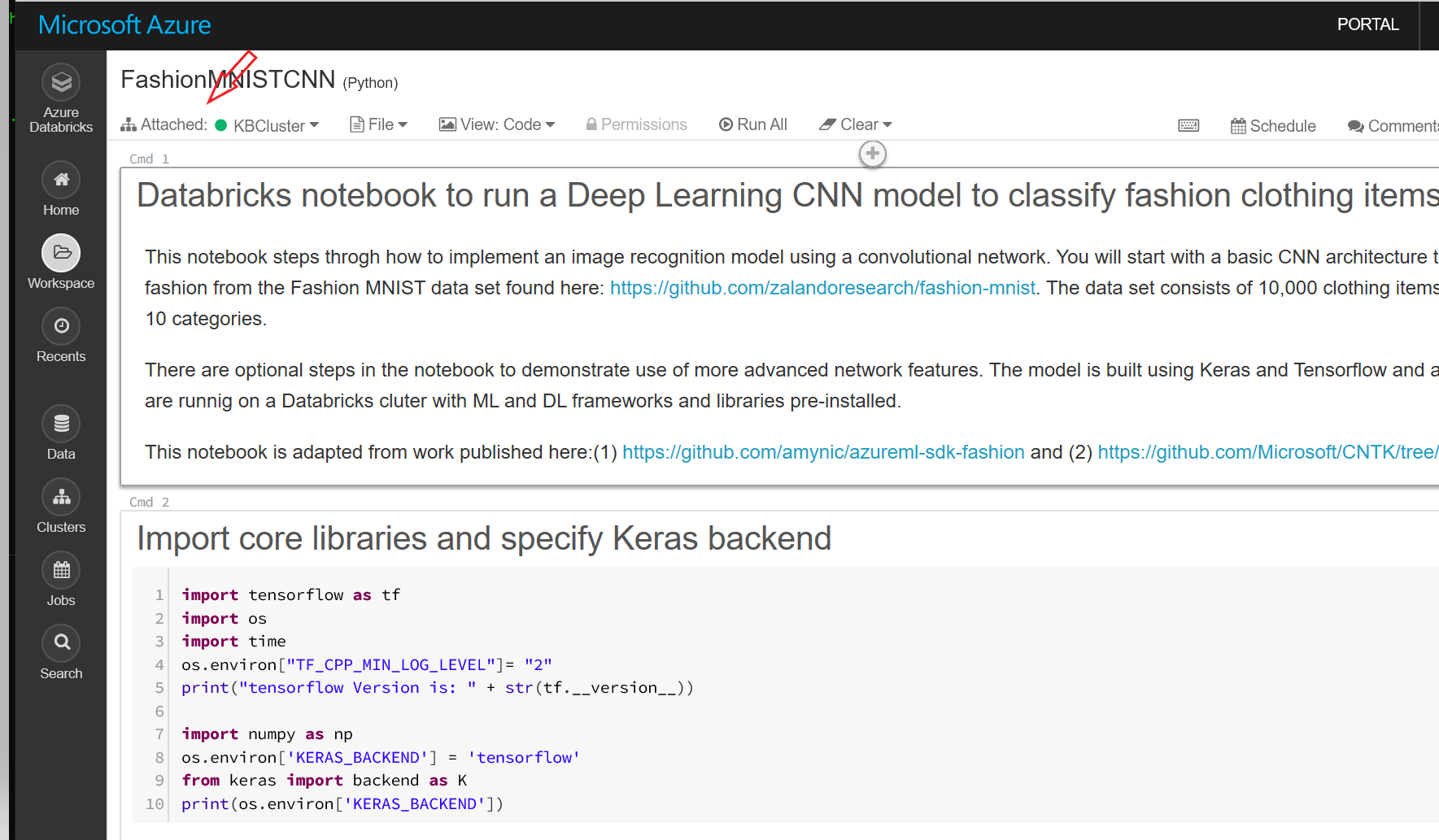
*Locate the* ***import*** *option in the dropdown*

1. Select the python notebook from the git hub repo under dnnworkshop3 -> labs -> 3.Keras-CNN -> and drag Fasion*MNISTCNN.ipynb* into the **Import Notebook** box and click **Import**:



*Import notebook into Databricks workspace*

1. Navigate back to the *Databricks* Workspace, select the notebook *FashionMNISTCNN* and attach it to the cluster:



*Attach notebook to the cluster*

You are now ready to work with the notebook.

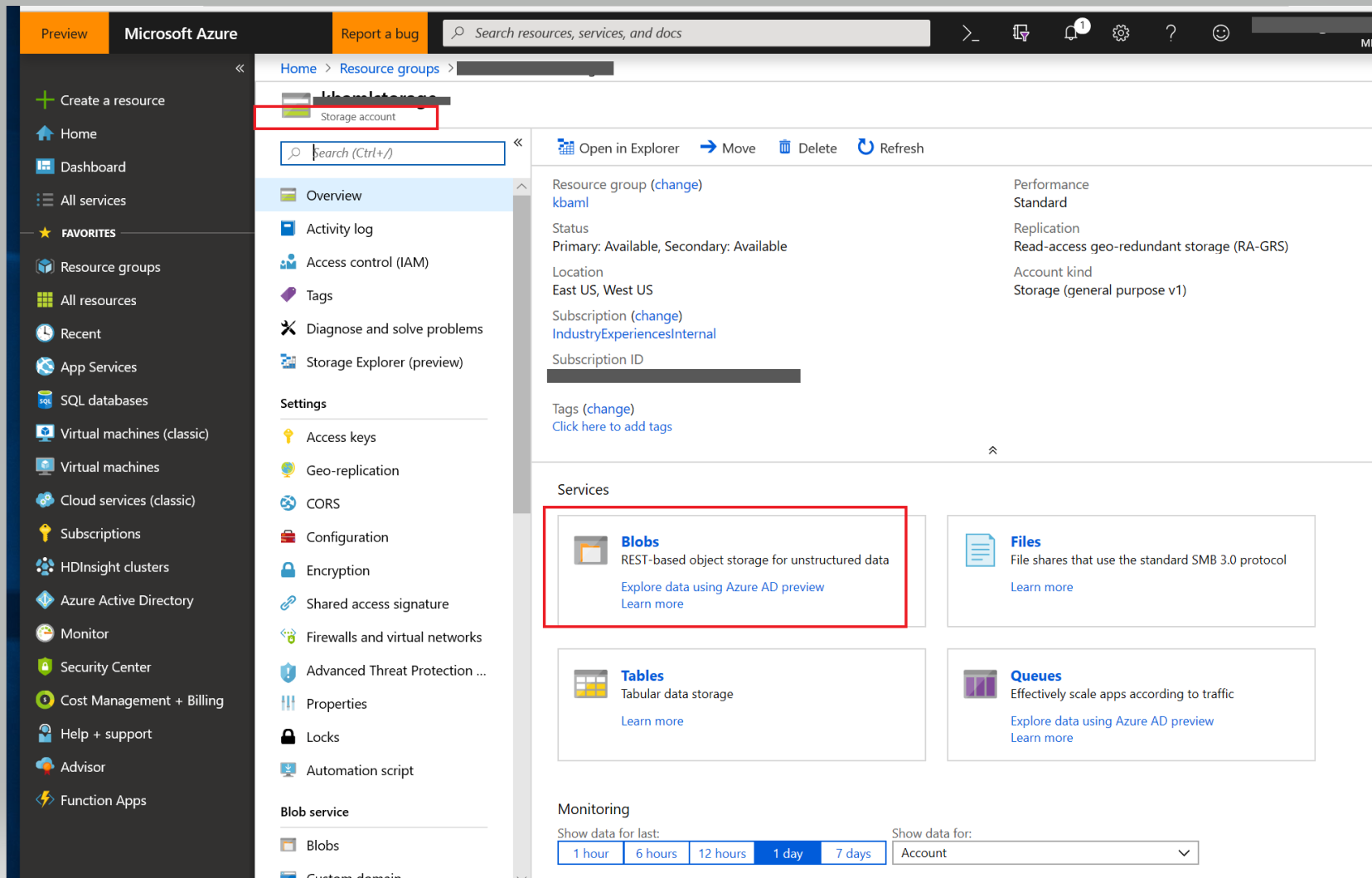
**Exercise 6 (Optional): Setup up storage access**

Some of the labs may require data be uploaded into azure blob storage in order to be access from the notebooks. If a datafile is more than 100GB in size, this may be the case and you will need to:

* Create an Azure storage account and a blob service container
* Upload the data into the blob container
* Create a mount point on the cluster to the azure blob container

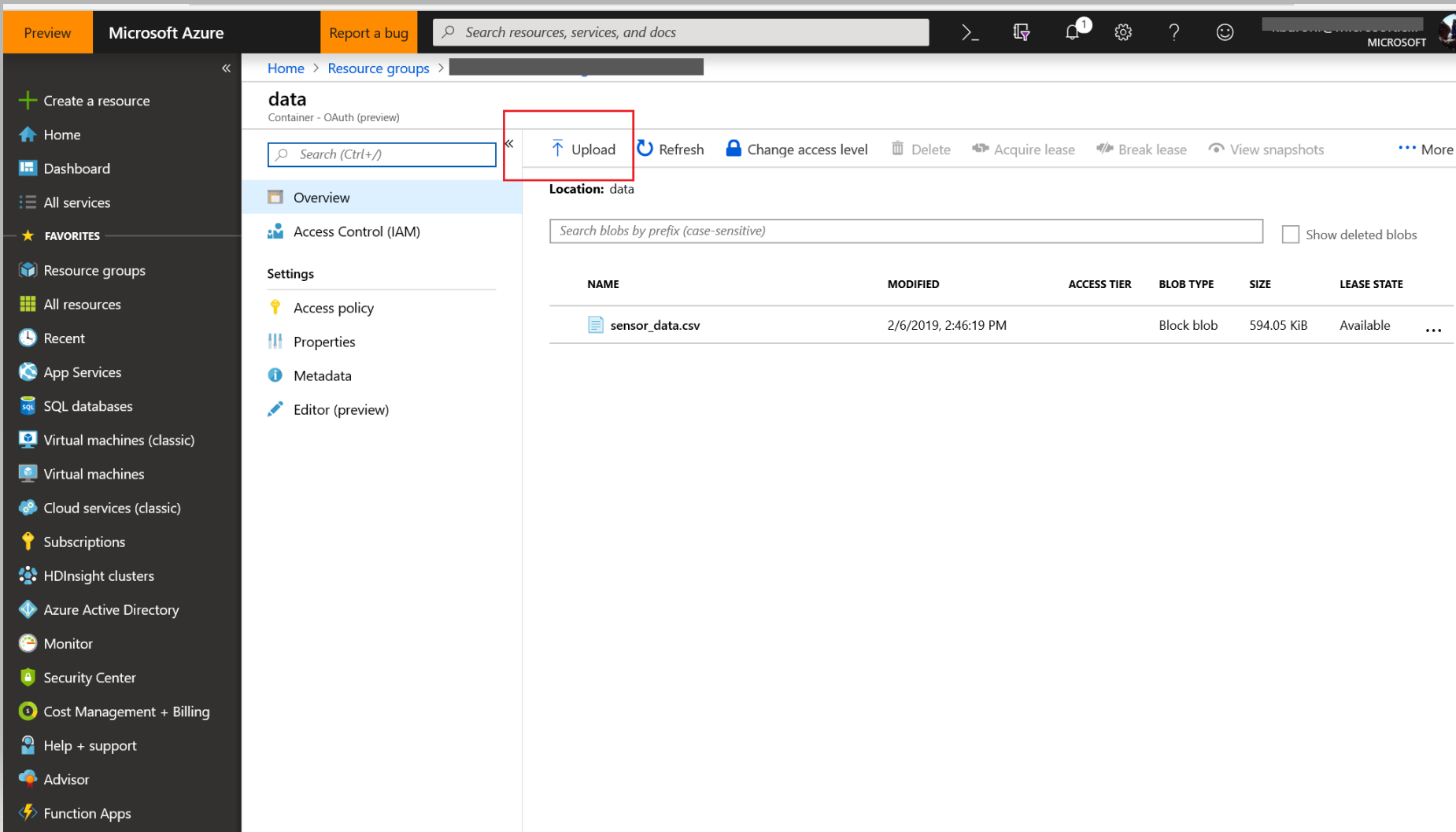
In this exercise, you will create a mount point from the *Databricks* cluster to azure storage blob. To test that storage access is correct, you will use the *usd\_to\_eur* csv file stored in the local github for the workshop.

1. Navigate to the portal and access the blob container in your azure storage account.



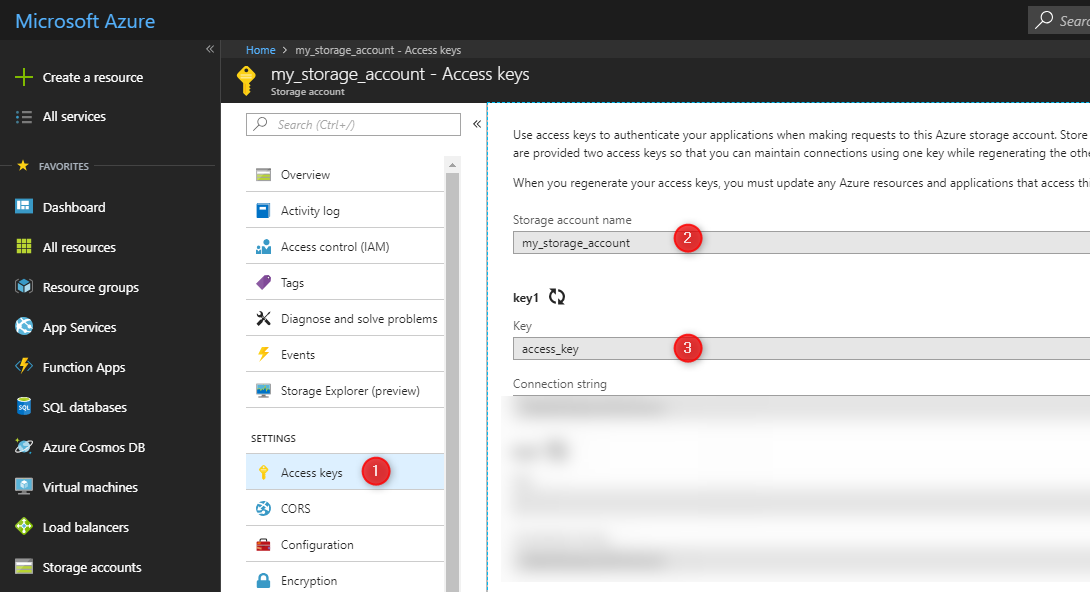
*Select blob storage*

1. Once in the target blob container, select **upload** and upload the csv file named *usd\_to\_eur* from the github repo location: dnnworkshop3 -> labs -> 1.Setup.



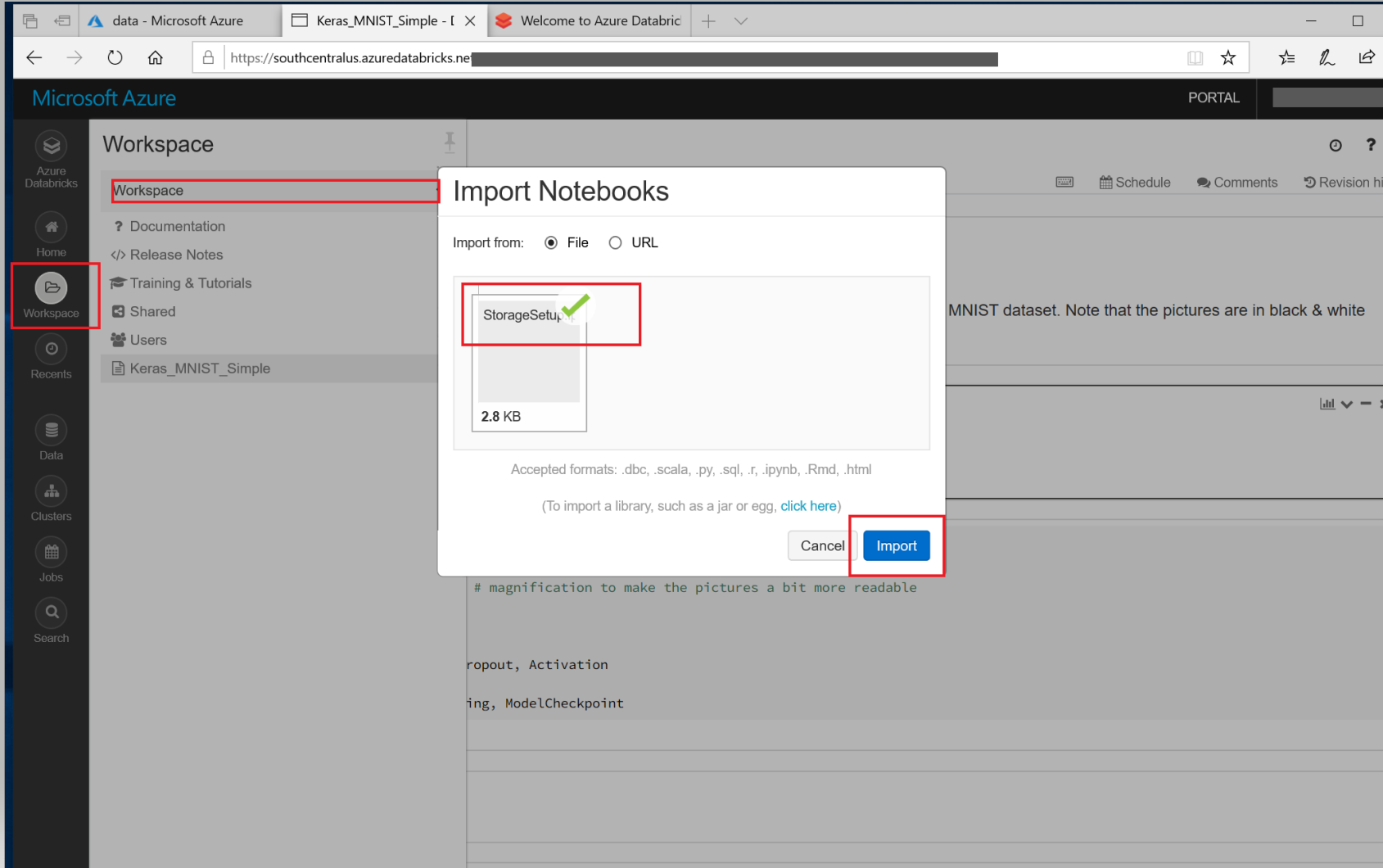
*Upload csv file into blob storage container*

1. Once the file is loaded, copy out the account name, access key, and blob container name. Later in this exercise you will need to copy these values into your notebook.



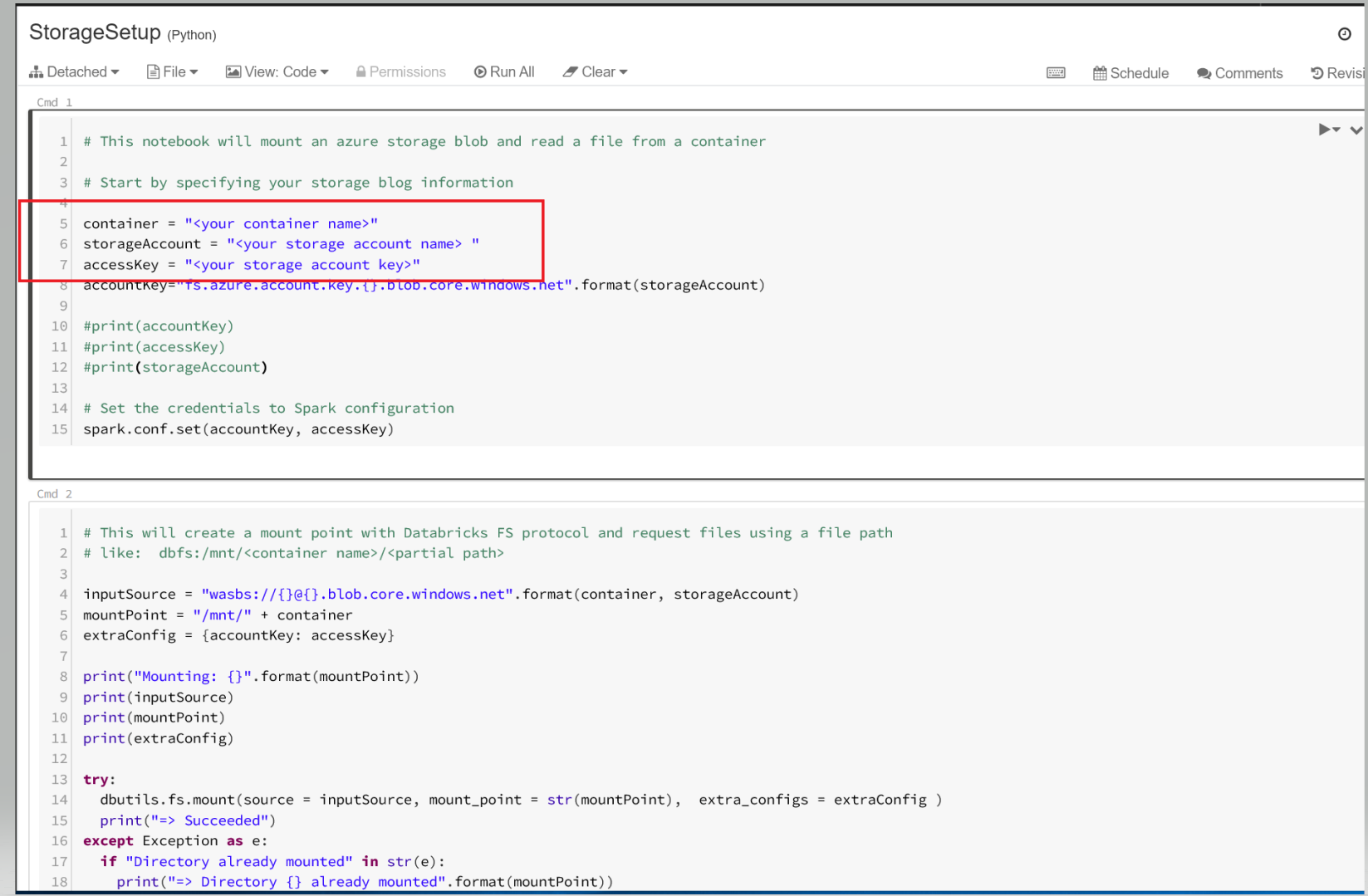
*Copy storage account name, access key, and blob container name*

1. Navigate back to your Databricks workspace and upload the notebook named StorageSetup.ipynb into your workspace. You can find the notebook in dnnworkshop3 -> labs -> 1.Setup.



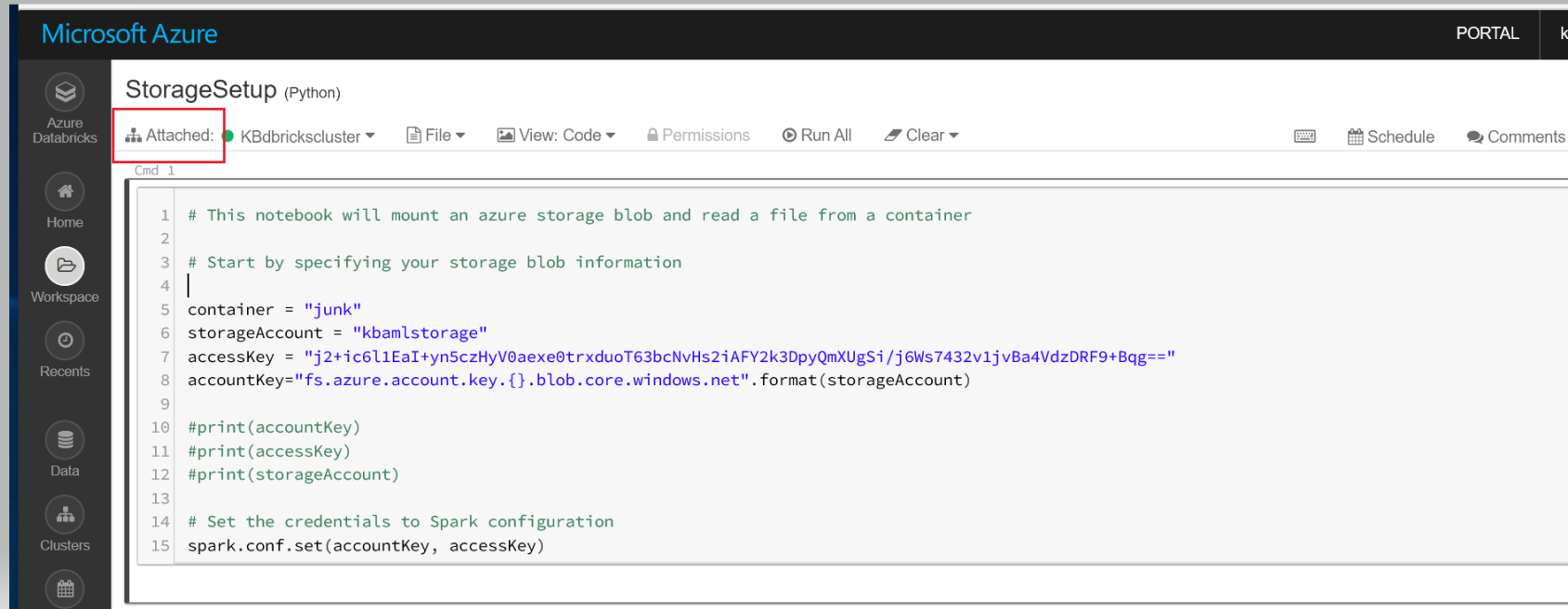
*Import StorageSetup.ipynb into workspace*

1. Navigate back to the workspace and click on *StorageSetup* to bring up the notebook. Substitute the values you copied from Step 3 into the correct cell I the notebook:



*Substitute in your storage account information*

1. Make sure the notebook is attached to the cluster



*Attach the notebook to the cluster*

1. Step through the notebook executing each of the cells to verify the blob container is successfully mounted on the cluster.

Your environment is now configured so you can load data into your azure blob container and access it directly through your notebooks.

**Spark Glossary**

**Spark context:** *Spark Context* is an object that tells Spark how and where to access a cluster. In a *Databricks* notebook, the *Spark Context* is already defined as a global variable **sc**.

**Spark session:** *Spark Session* is the entry point for reading data and execute SQL queries over data and getting the results. *Spark Session* can also be used to set runtime configuration options. In a *Databricks* notebook, the *Spark session* is already defined as a global variable **spark**.

# Databricks Notebooks have some Apache Spark variables already defined.

# SparkContext: sc

# SQLContext/HiveContext: sqlContext

# SparkSession (Spark 2.x): spark

print("Spark version", sc.version, spark.sparkContext.version, spark.version) print("Python version", sc.pythonVer)

# Spark version 2.3.0 2.3.0 2.3.0

# Python version 3.5

**RDD:** Resilient Distributed Dataset (RDD)

**Dataframe (DF):** A **DataFrame** is a distributed collection of rows under **named columns**. It is the same as a table in a relational database. It is close to [*Pandas DataFrames*](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html). A DataFrame has the ability to handle petabytes of data and is built on top of RDDs. A DataFrame is mapped to a **relational schema.**

**Dataset:** A Dataset is a *strongly-typed* DataFrame. A DataFrame as an *alias* for a collection of generic objects *Dataset[Row]*, where a *Row* is a generic *untyped* JVM object. *DataFrame* and *Dataset* are now merged in an unified APIs in Spark 2.0. Learn more on the differences between DF, Dataset, and RDD with this [link](https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html) from Databricks blog.

**Python or Scala notebooks?** If you are familiar with Python, you can stay with Python as you can do almost everything in Python (the Python library is PySpark). However, Scala is the native language for Spark and because Spark is written in Scala so you will find 80% of the examples, libraries, and discussions in StackOverflow in Scala. The good news is that within a *Databricks* notebook, you can mix multiple language access.