Data Engineering in the Cloud

Introduction to Azure Databricks

Xuemao Zhang
East Stroudsburg University

January 18, 2025

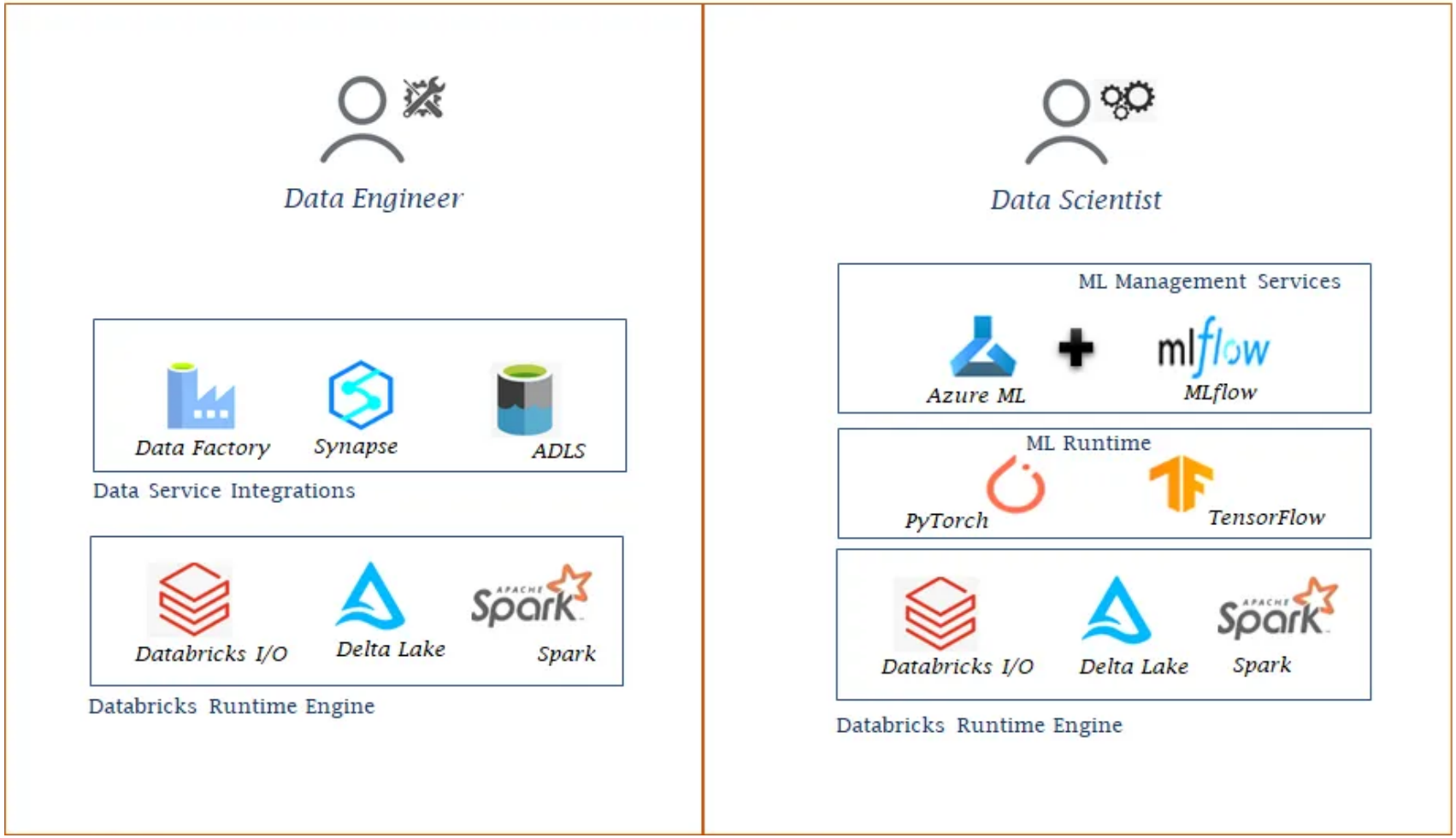
Outline

- Introduction to Azure Databricks
- Databricks File System
- Lab 1: Creating an Azure Databricks Workspace
- Lab 2: Creating a Cluster
- Lab 3: Using DataFrames in Azure Databricks

Introduction to Azure Databricks

- Azure Databricks is a fast, easy, and collaborative **Apache Spark-based** analytics platform optimized for Microsoft Azure.
- Key Features
  - Collaborative Workspaces: Interactive environments for team collaboration
  - Automated Cluster Management: Easy scaling and management of Spark clusters
  - Machine Learning: Built-in tools for developing and deploying models
  - Seamless Integration: Works with Azure services like Synapse Analytics, Data Lake Storage, and Power BI

Introduction to Azure Databricks



Reading and Writing Data in Azure Databricks

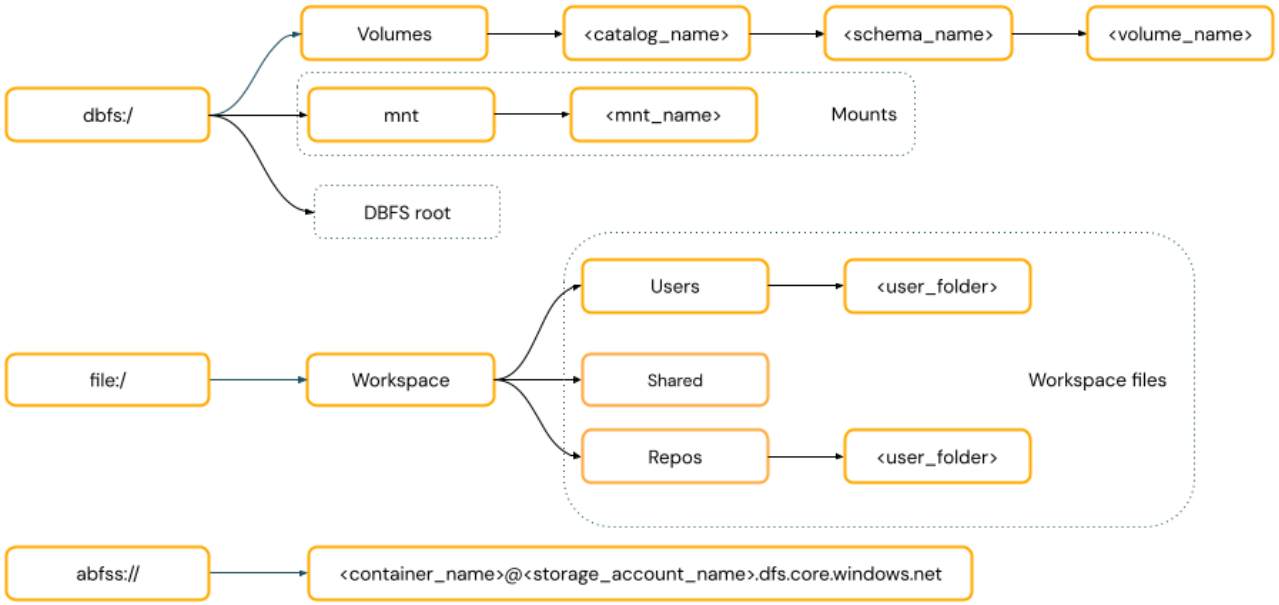
- Several types of data can be read with Azure Databricks
- Reading Data in CSV Format
- Reading Data in JSON Format
  - `val df = spark.read.json("example.json")`
  - `val mdf = spark.read.option("multiline","true").json("/tmp/demo.json")`
    `mdf.show(false)`
- Reading Data in Parquet Format
  - `data = spark.read.parquet("/tmp/Parquetfile")`

Reading and Writing Data in Azure Databricks

- Writing data: Apache Spark can be used to write data frame into disk and data into multiple files.
  - `df.write.format("csv").mode("overwrite").save("/my-file/demo csv")`

Databricks File System

- Databricks file system, also known as DBFS, is a distributed file system.
  - It has a master slave architecture to attain reliability.
  - https://learn.microsoft.com/en-us/azure/databricks/files/
- URI-style paths include a URI scheme.



Databricks File System

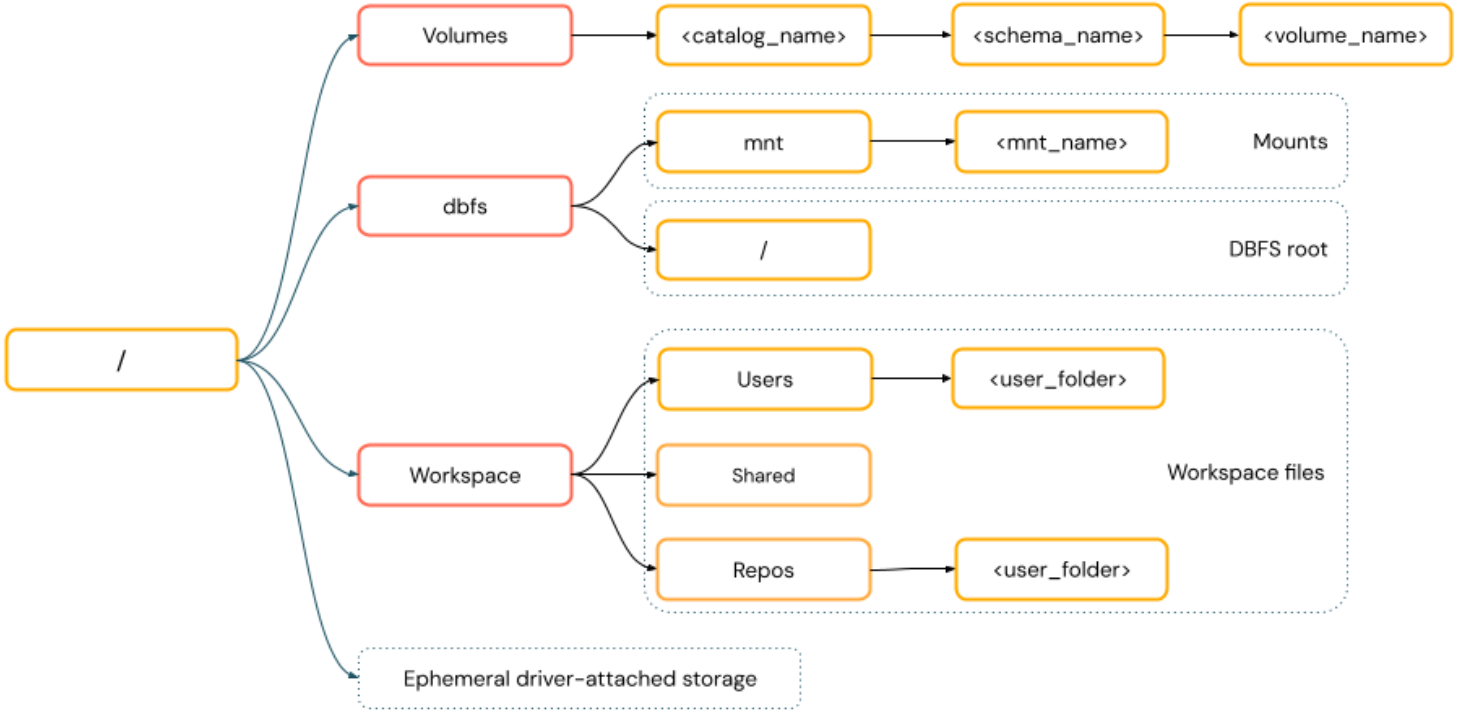
- The structure shows how different storage paths are organized and accessed in Azure Databricks.
- `dbfs:/`
  - Volumes: Structured as `<catalog_name>/<schema_name>/<volume_name>`.
  - mnt: Mount points for external storage, structured as `<mnt_name>`.

file:/

- Workspace: Contains:
  - Users: Individual user folders.
  - Shared: Shared files.
  - Repos: Repositories, structured as `<user_folder>`.
- `abfss://`: Connects to Azure Blob Storage using the format `<container_name>@<storage_account_name>.dfs.core.windows.net`.

Databricks File System

- POSIX-style paths provide data access relative to the driver root (/)



Lab 1: Creating an Azure Databricks Workspace

- First follow what we did before, create a Data Lake Storage account and upload the data `population.csv` to the storage

Create a storage account ...

Basics | Advanced | Networking | Data protection | Encryption | Tags | Review + create

View automation template

Basics

Subscription	Azure for Students
Resource group	NetworkWatcherRG
Location	East US
Storage account name	xzhang2storage
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

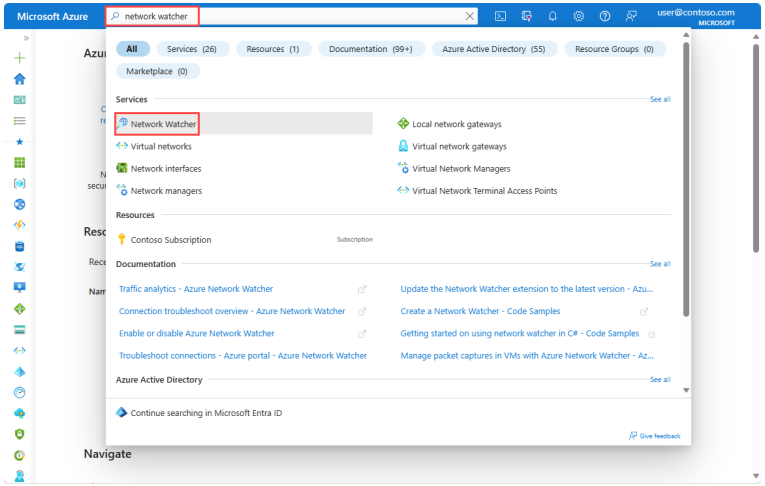
Enable hierarchical namespace	Enabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

Security

Secure transfer	Enabled
Blob anonymous access	Disabled
Allow storage account key access	Enabled
Default to Microsoft Entra authorization in the Azure portal	Disabled

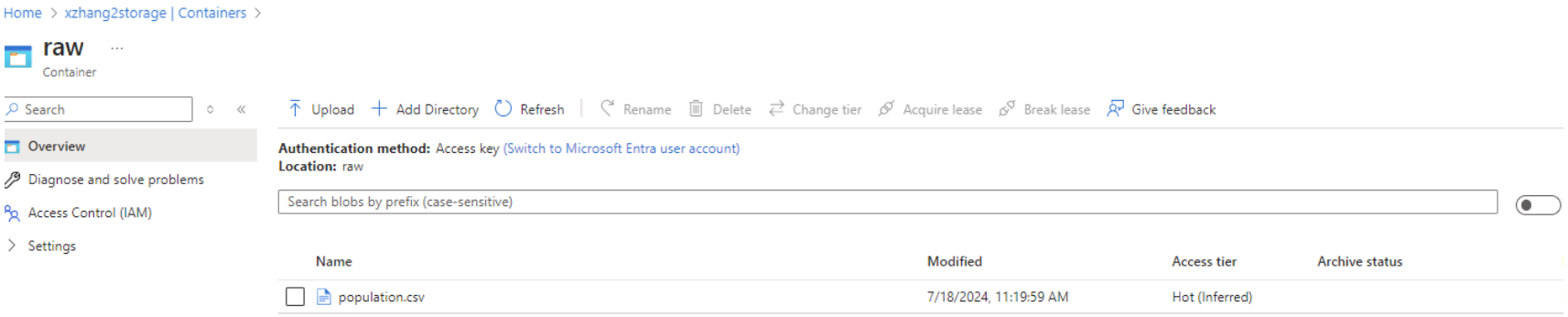
Lab 1: Creating an Azure Databricks Workspace

- `Network Watcher` is a regional service that enables you to monitor and diagnose conditions at a network scenario level in, to, and from Azure. - It includes a suite of tools for monitoring, diagnosing, viewing metrics, and enabling or disabling logs for resources in a virtual network.
- This resource group is automatically generated by Azure when you enable Network Watcher in a region.
  - https://learn.microsoft.com/en-gb/azure/network-watcher/network-watcher-overview
- When you enable Network Watcher, Azure creates a resource group named NetworkWatcherRG in each region where it is enabled.



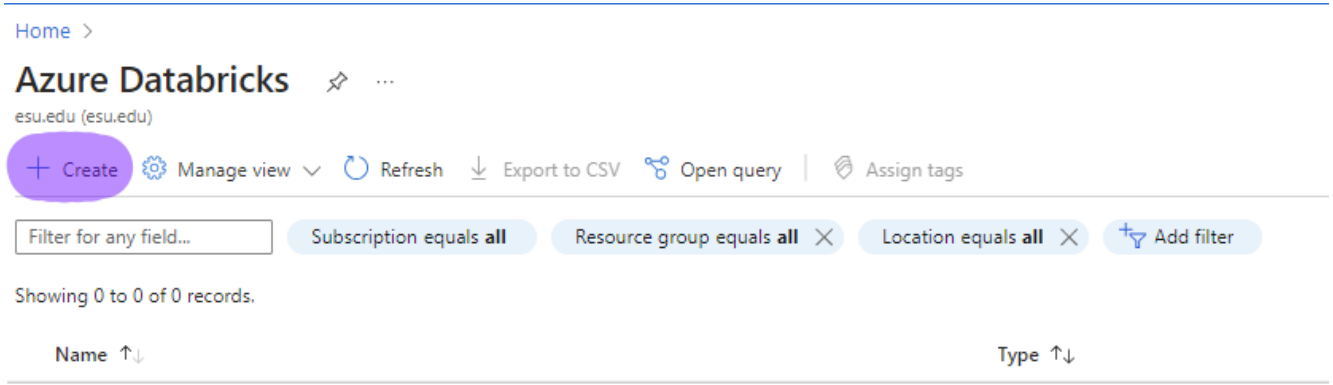
Lab 1: Creating an Azure Databricks Workspace

- First follow what we did before, create a Data Lake Storage account and upload the data `population.csv` to the storage



Lab 1: Creating an Azure Databricks Workspace

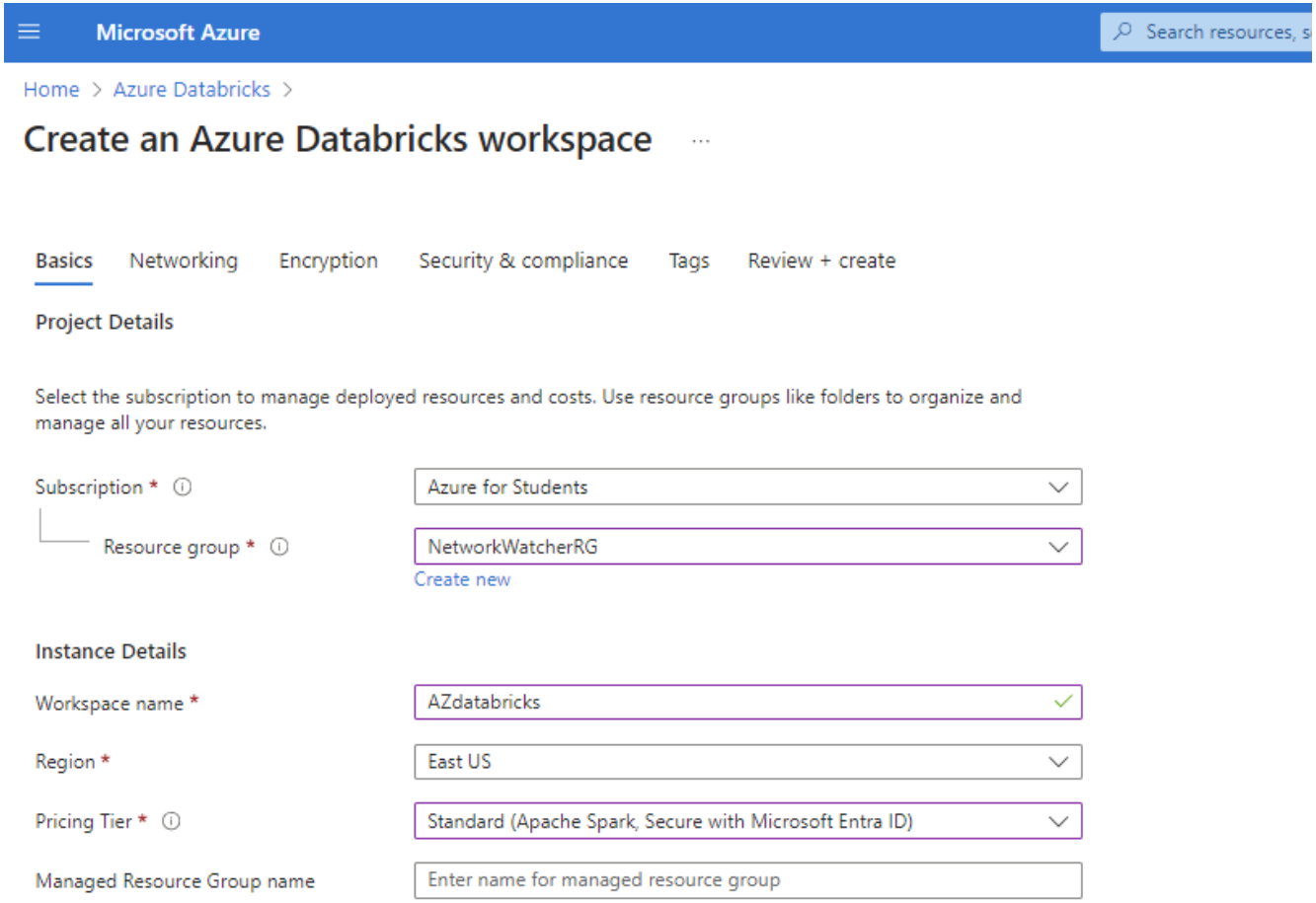
- Search for `Azure Databricks` in the search bar and click on the `Create`



No azure

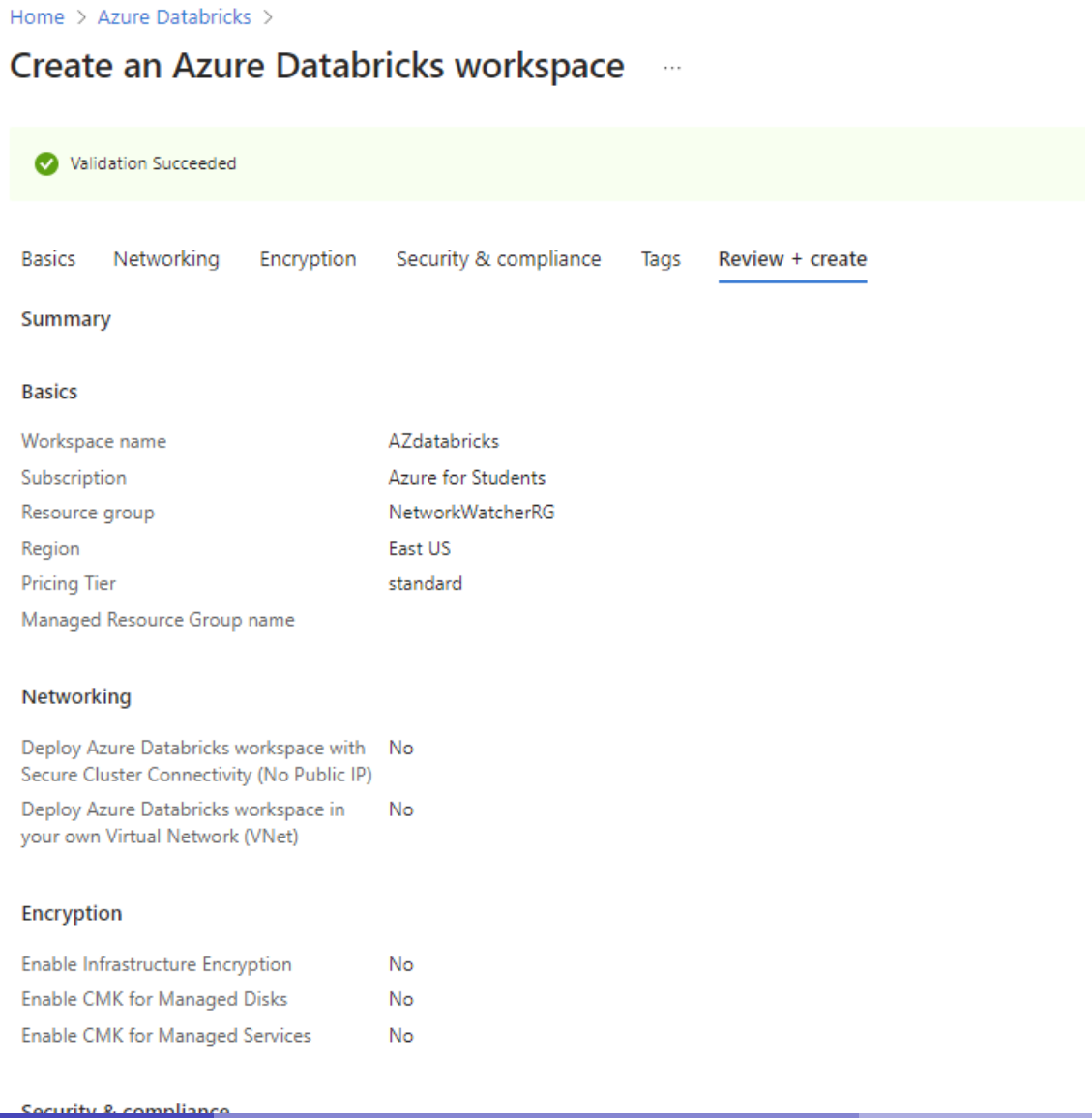
Lab 1: Creating an Azure Databricks Workspace

- Specify workspace name, region, and pricing tier as shown below:



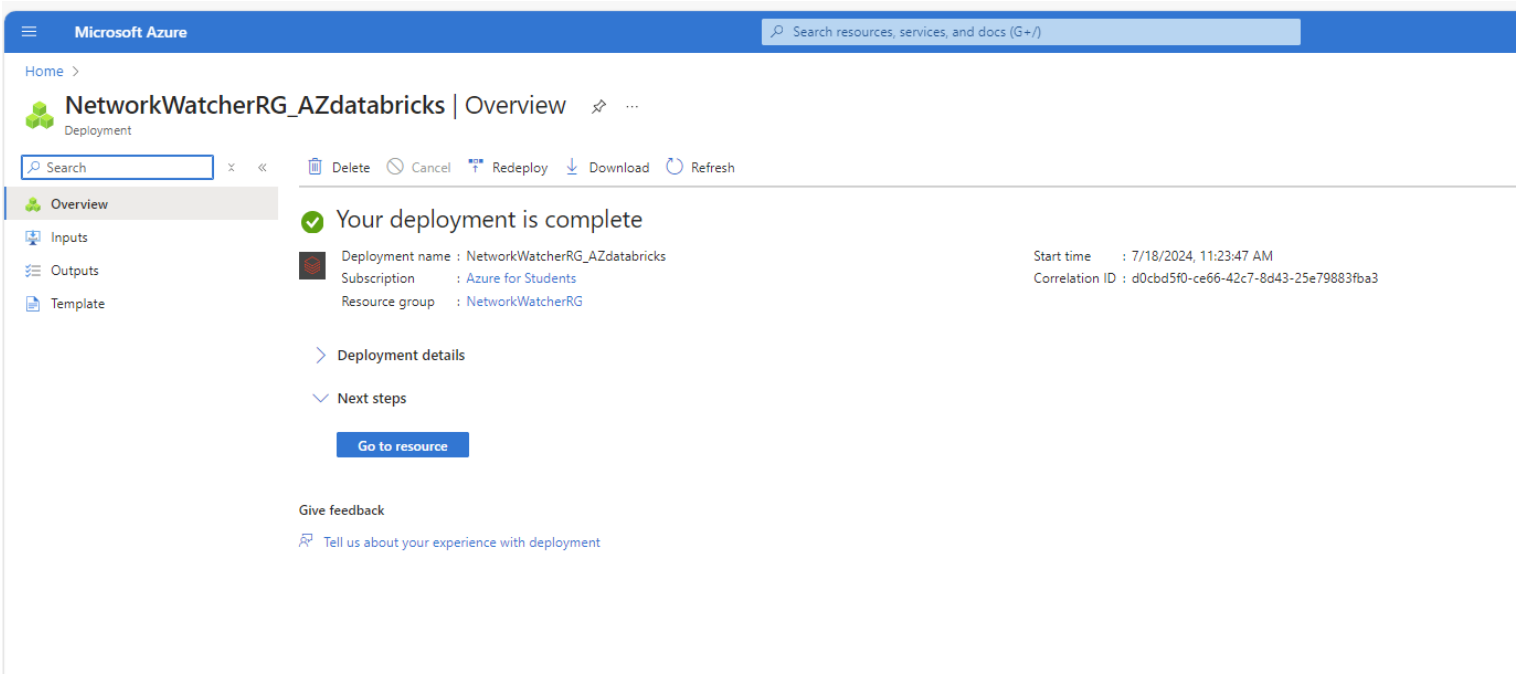
Lab 1: Creating an Azure Databricks Workspace

- We can leave the other settings to default and go to and click `review + create`



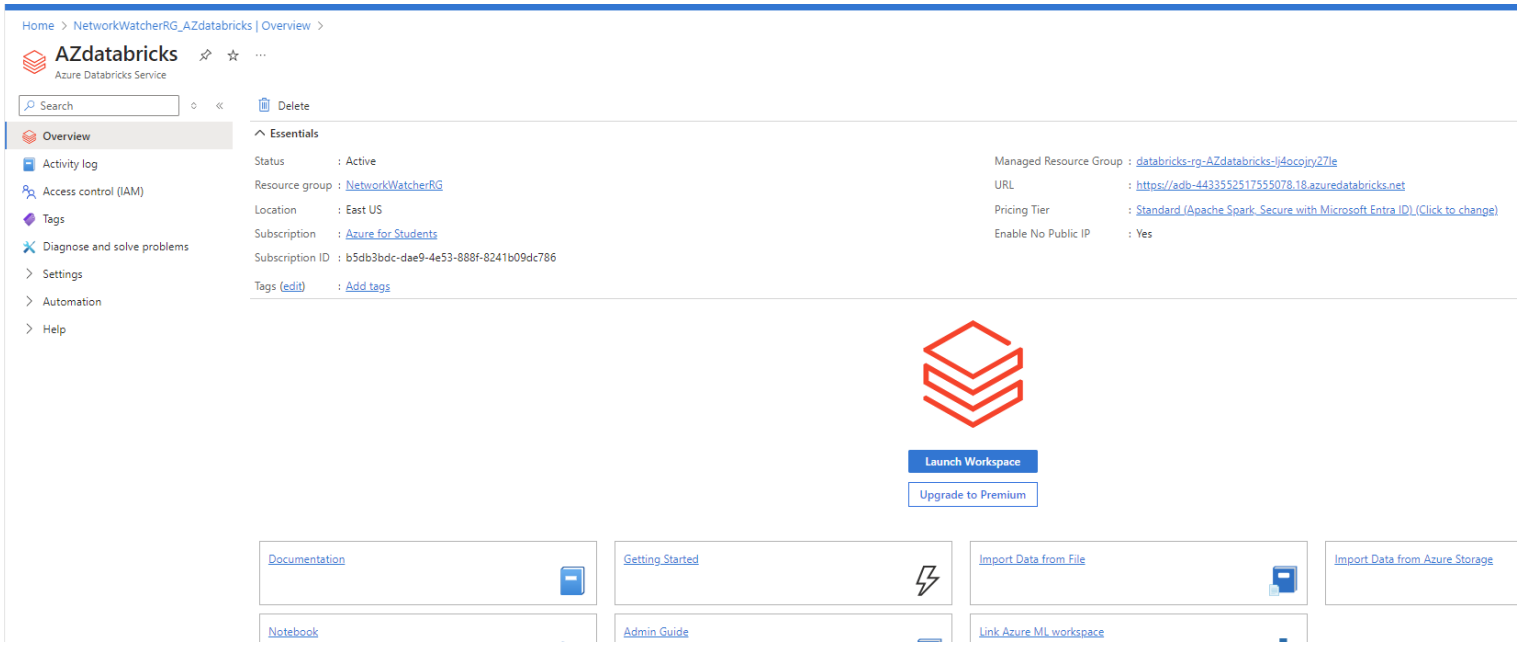
Lab 1: Creating an Azure Databricks Workspace

- Once the deployment is completed, click on the `Go to resource` button



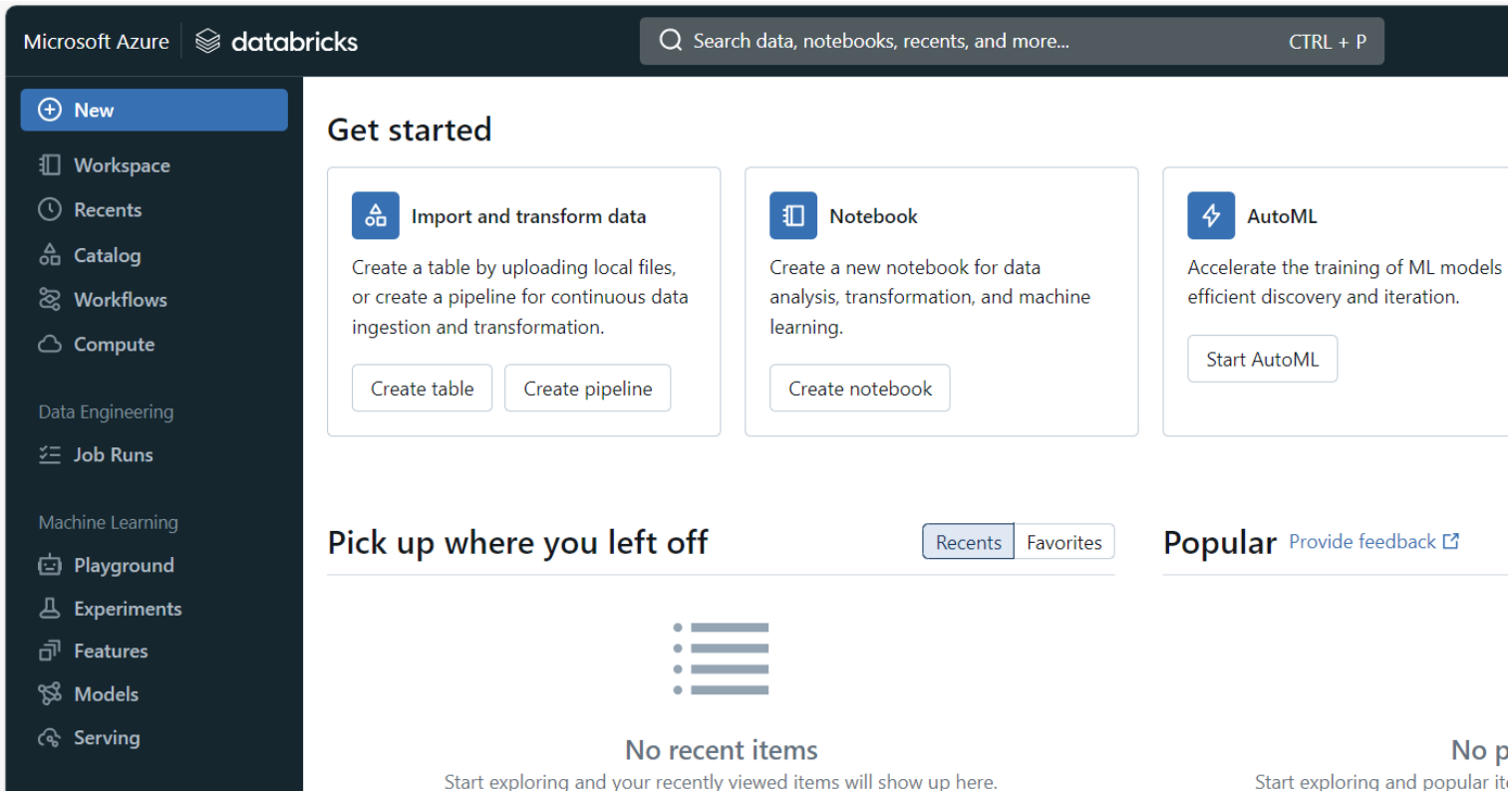
Lab 1: Creating an Azure Databricks Workspace

- Click on the "Launch workspace" button to launch the workspace



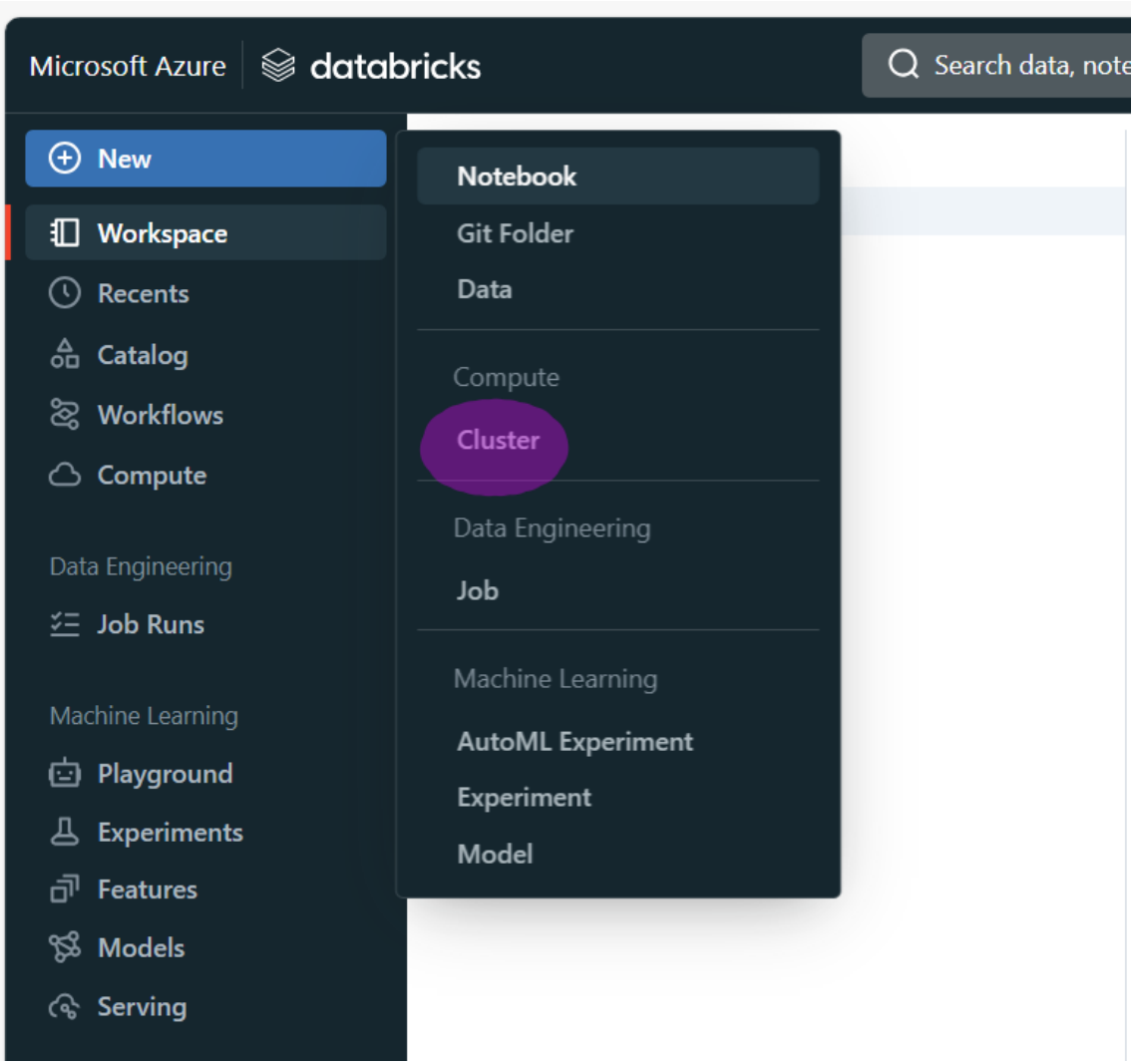
Lab 1: Creating an Azure Databricks Workspace

- Thi is how the Azure Databricks workspace looks like



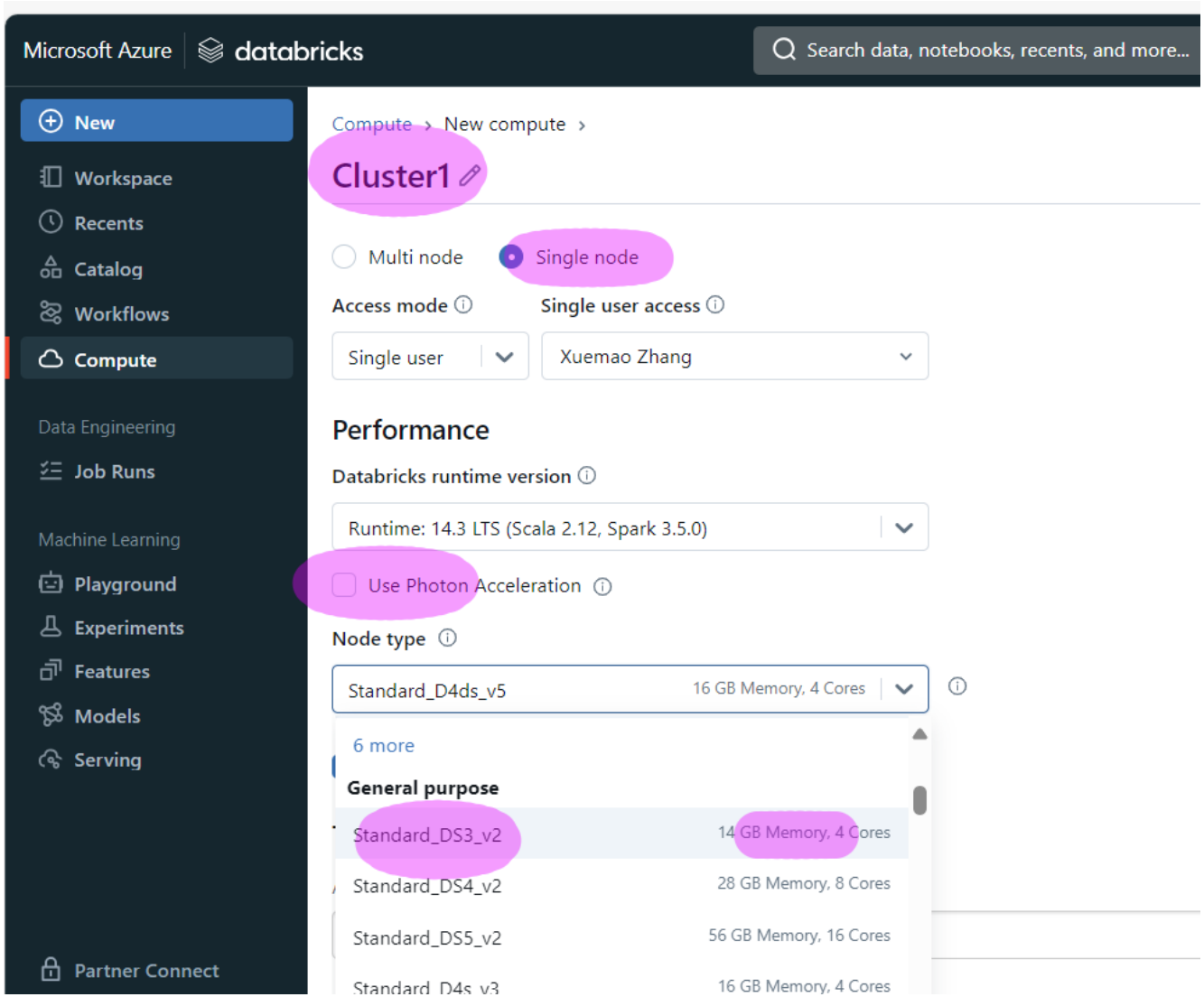
Lab 2: Creating a Cluster

- Databricks eliminate the complexity and provide convenience to get a spark cluster. The Spark experience is seamless and requires no management.
- Click on `Cluster` under the menu `+ New` to create a cluster



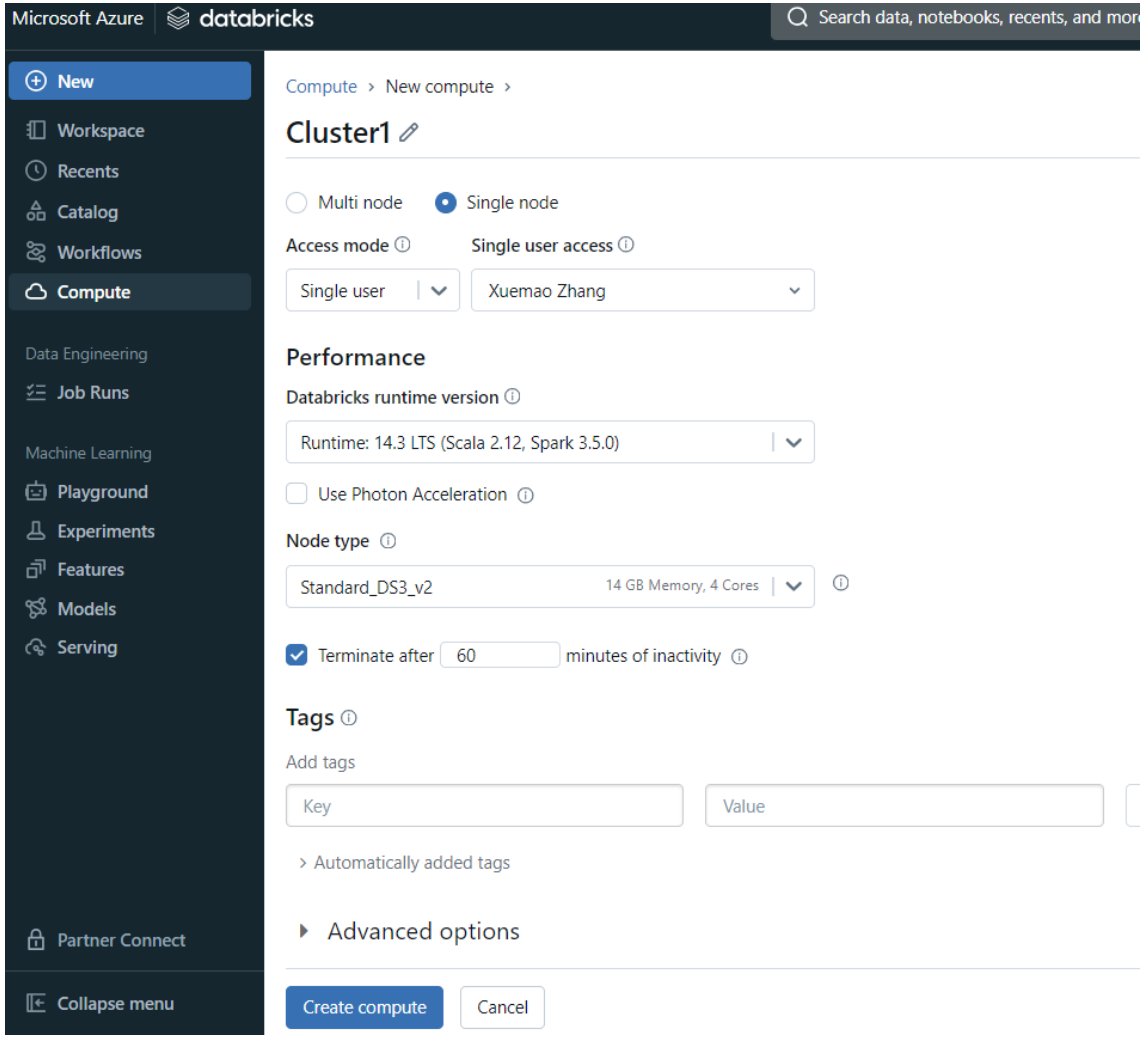
Lab 2: Creating a Cluster

- Specify the name for the "Cluster"; and Select the Cluster mode



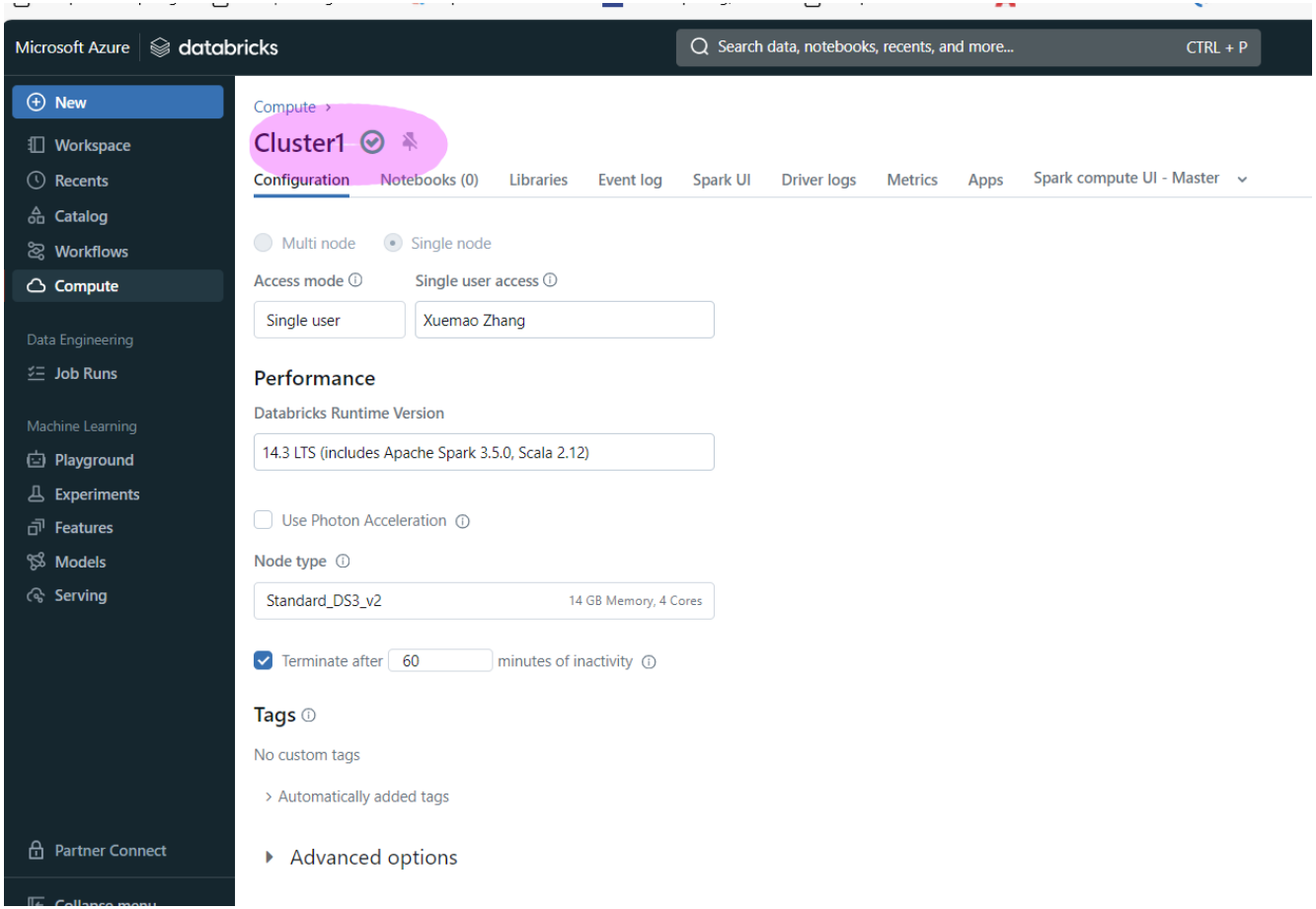
Lab 2: Creating a Cluster

- We can specify the termination time as well. The defaulted value is 120 minutes
  - And then click on the `Create compute` button



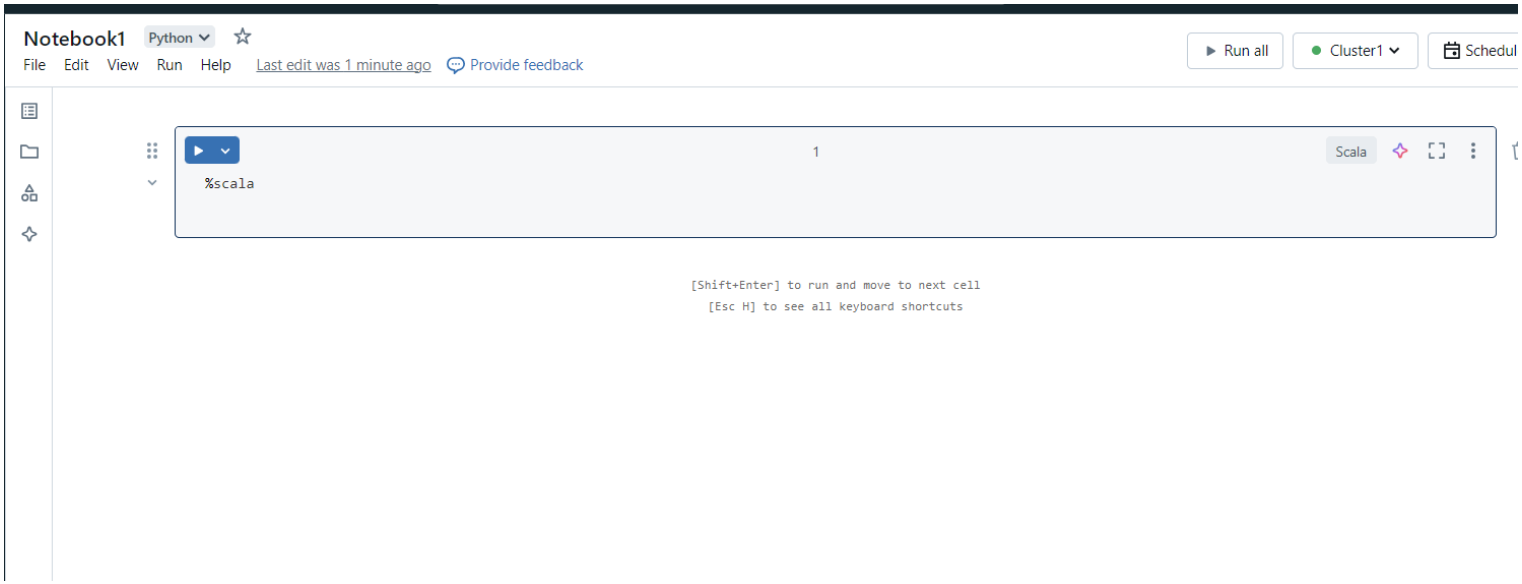
Lab 2: Creating a Cluster

- If successful, you should be able to see this



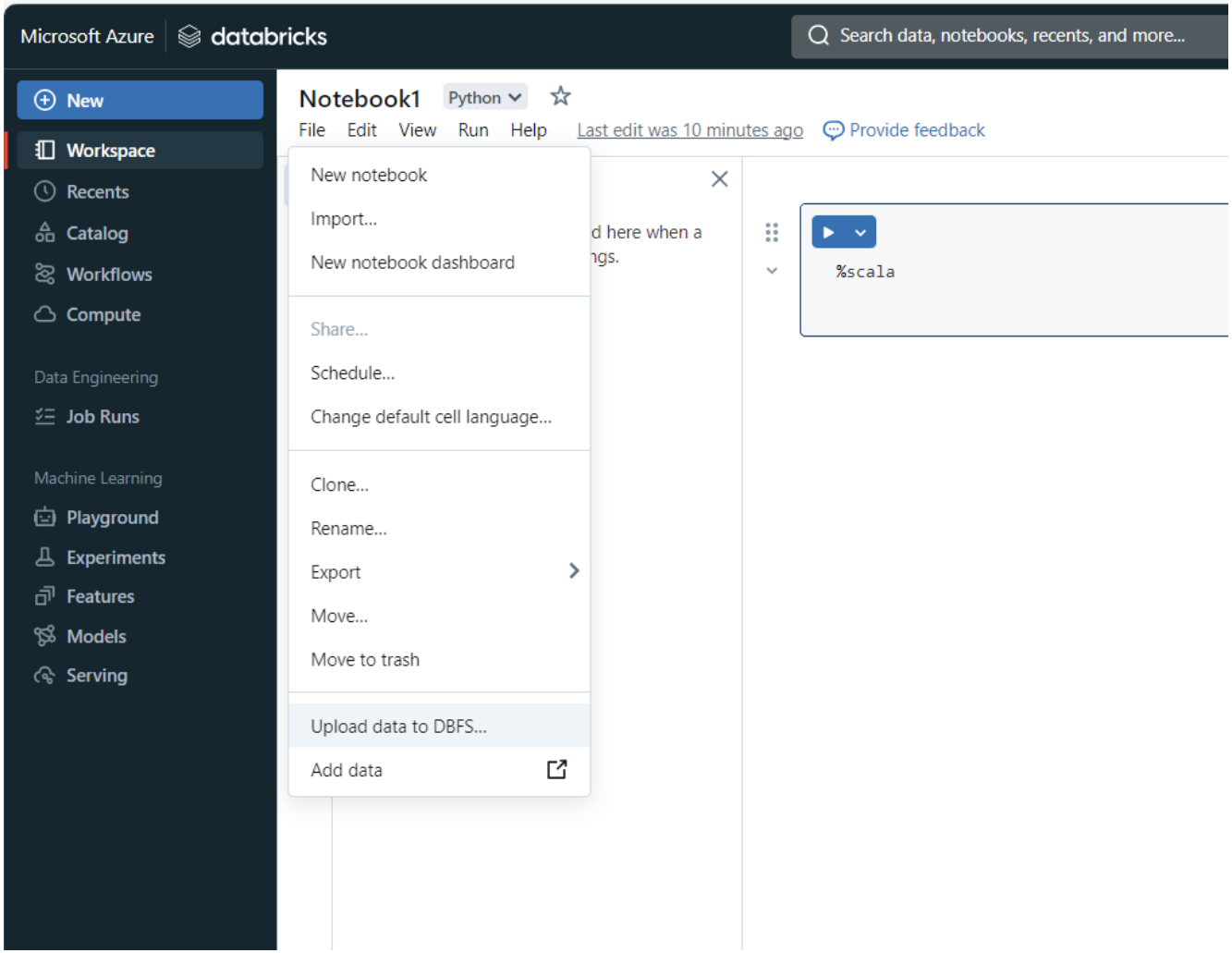
Lab 3: Using DataFrames in Azure Databricks

- Now create a notebook in the cluster and rename the notebook

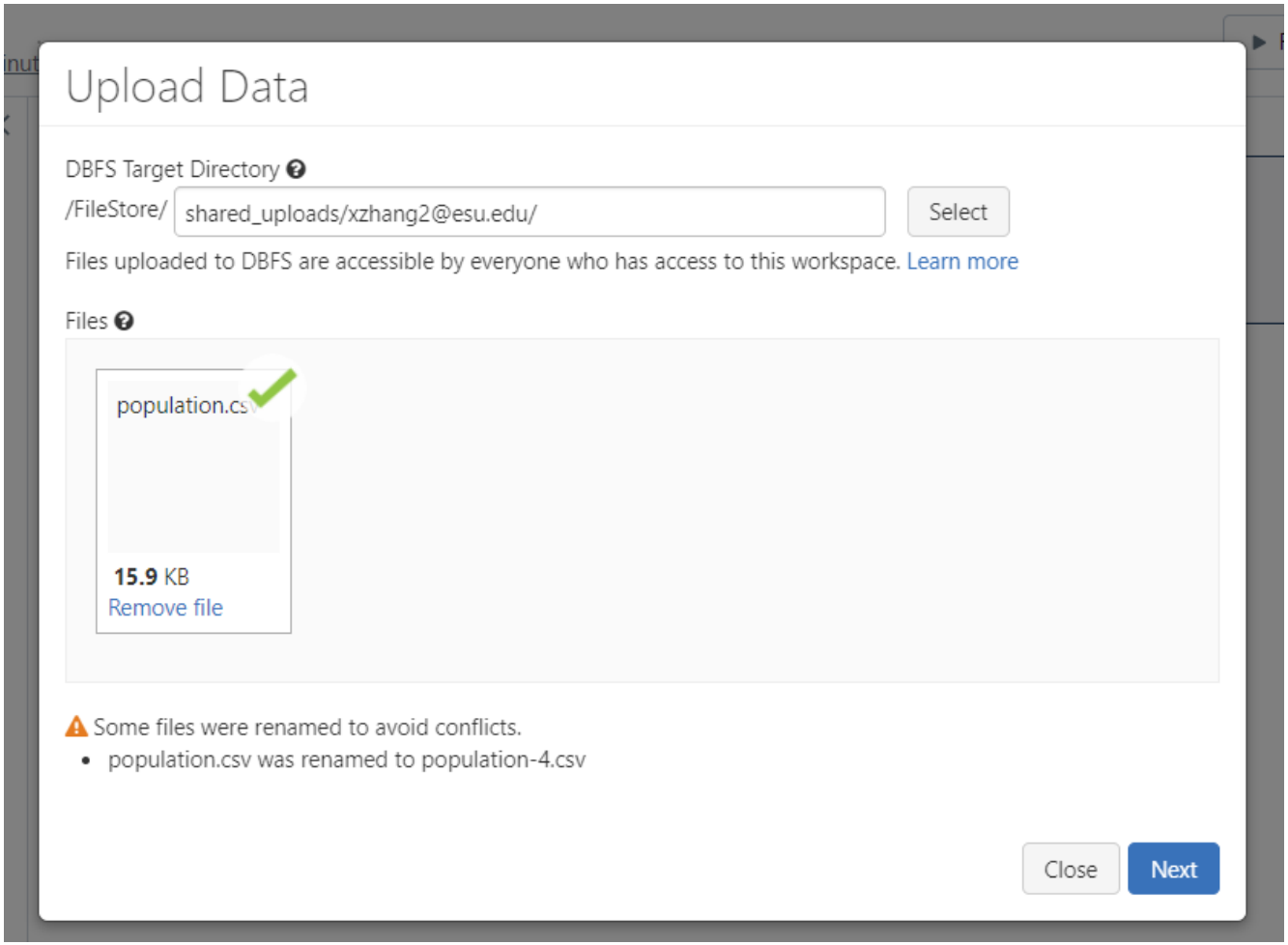


Lab 3: Using DataFrames in Azure Databricks

- In the menus of the notebook, Click `file` → `Upload data to DBFS...` to upload the file `population.csv` on your local disk
  - We consider how to get the data from storage later



Lab 3: Using DataFrames in Azure Databricks

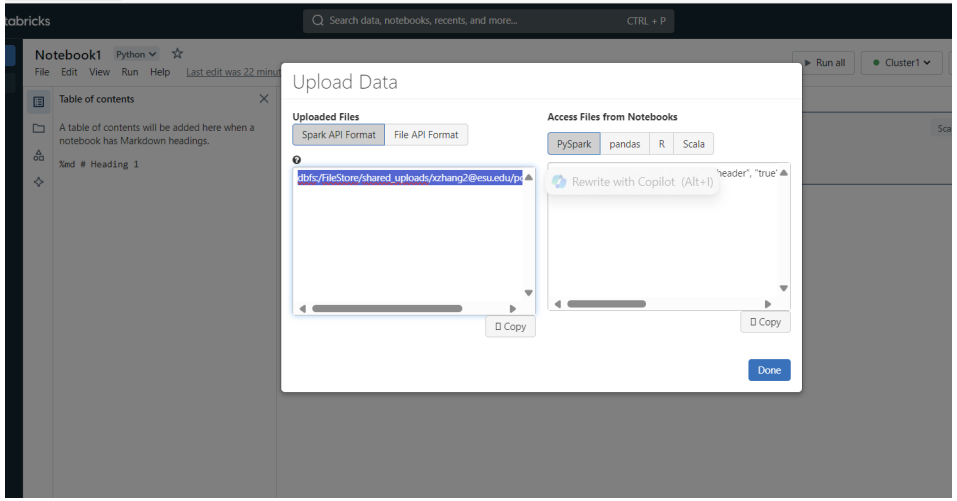


Lab 3: Using DataFrames in Azure Databricks

- Copy the code from the two boxes

```
dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv
```

```
df1 = spark.read.format("csv").
option("header", "true").
load("dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv")
```

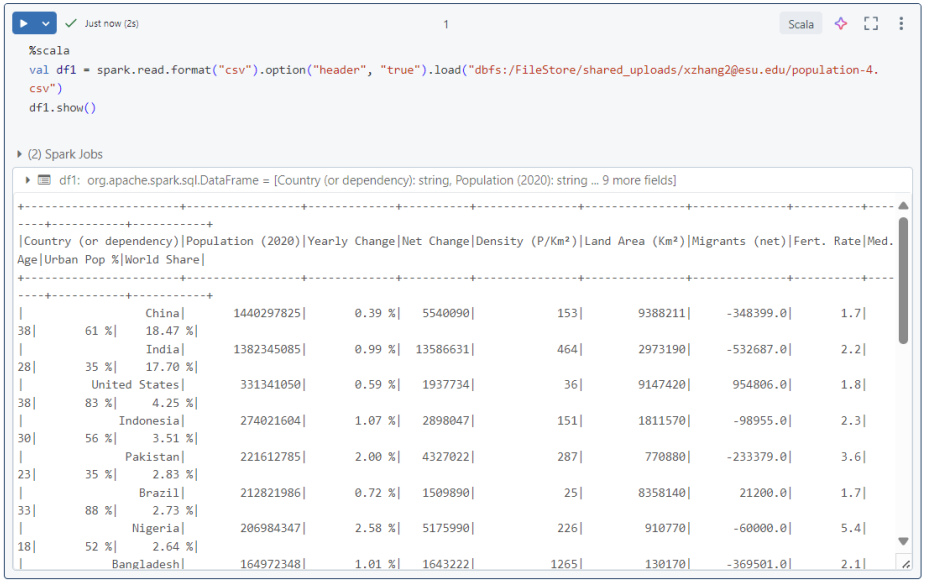


- Click on `Done`.

Lab 3: Using DataFrames in Azure Databricks

- Paste the code to the notebook

```
val df1 = spark.read.format("csv").
option("header", "true").
load("dbfs:/FileStore/shared_uploads/xzhang2@esu.edu/population-4.csv")
df1.show()
```



Lab 3: Using DataFrames in Azure Databricks

- The DataFrame can also be displayed by executing the display() function as shown below.

Just now (1s) 2 Scala

```
%scala
display(df1)
```

(1) Spark Jobs

Table

	Country (or dependency)	Population (2020)	Yearly Change	Net Change	Density (P/Km²)	Land
1	China	1440297825	0.39 %	5540090	153	9388211
2	India	1382345085	0.99 %	13586631	464	2973190
3	United States	331341050	0.59 %	1937734	36	9147420
4	Indonesia	274021604	1.07 %	2898047	151	1811570
5	Pakistan	221612785	2.00 %	4327022	287	770880
6	Brazil	212821986	0.72 %	1509890	25	8358140
7	Nigeria	206984347	2.58 %	5175990	226	910770
8	Bangladesh	164972348	1.01 %	1643222	1265	130170
9	Russia	145945524	0.04 %	62206	9	16376870
10	Mexico	129166028	1.06 %	1357224	66	1943950
11	Japan	126407422	-0.30 %	-383840	347	364555
12	Ethiopia	115434444	2.57 %	2884858	115	1000000
13	Philippines	109830324	1.35 %	1464463	368	298170
14	Egypt	102659126	1.94 %	1946331	103	995450

Lab 3: Using DataFrames in Azure Databricks

- Study the underlying data types by executing the printSchema() function

Just now (<1s) 3 Scala

```
%scala
df1.printSchema()
```

```
root
 |-- Country (or dependency): string (nullable = true)
 |-- Population (2020): string (nullable = true)
 |-- Yearly Change: string (nullable = true)
 |-- Net Change: string (nullable = true)
 |-- Density (P/Km²): string (nullable = true)
 |-- Land Area (Km²): string (nullable = true)
 |-- Migrants (net): string (nullable = true)
 |-- Fert. Rate: string (nullable = true)
 |-- Med. Age: string (nullable = true)
 |-- Urban Pop %: string (nullable = true)
 |-- World Share: string (nullable = true)
```

Lab 3: Using DataFrames in Azure Databricks

- To change the names of the columns in a DataFrame using Scala in Azure Databricks, you can use the `withColumnRenamed` method for each column you want to rename

```
%scala
val df2 = df1
  .withColumnRenamed("Country (or dependency)", "Country")
  .withColumnRenamed("Population (2020)", "Population_2020")
  .withColumnRenamed("Yearly Change", "Yearly_Change")
  .withColumnRenamed("Net Change", "Net_Change")
  .withColumnRenamed("Density (P/Km²)", "Density_PKm2")
  .withColumnRenamed("Land Area (Km²)", "Land_Area_Km2")
  .withColumnRenamed("Migrants (net)", "Migrants_net")
  .withColumnRenamed("Fert. Rate", "Fertility_Rate")
  .withColumnRenamed("Med. Age", "Median_Age")
  .withColumnRenamed("Urban Pop %", "Urban_Pop_Percent")
  .withColumnRenamed("World Share", "World_Share")

df2.printSchema()
```

Lab 3: Using DataFrames in Azure Databricks

- To filter the columns that are needed, execute the command given below.

Just now (1s) 5

```
%scala
df2.select("Country", "Population_2020").show()
```

(1) Spark Jobs

```
+-------------+---------------+
|      Country|Population_2020|
+-------------+---------------+
|        China|     1440297825|
|        India|     1382345085|
|United States|      331341050|
|    Indonesia|      274021604|
|     Pakistan|      221612785|
|       Brazil|      212821986|
|      Nigeria|      206984347|
|   Bangladesh|      164972348|
|       Russia|      145945524|
|       Mexico|      129166028|
|        Japan|      126407422|
|     Ethiopia|      115434444|
|  Philippines|      109830324|
|        Egypt|      102659126|
|      Vietnam|       97490013|
|     DR Congo|       90003954|
|       Turkey|       84495243|
```

Lab 3: Using DataFrames in Azure Databricks

- To filter the row that contains the data for India, execute the filter function given below.

Just now (1s) 6 Scala

```
%scala
df2.filter(df2("Country")=="India").show()
```

(1) Spark Jobs

```
+-------+---------------+-------------+----------+------------+-------------+------------+-----
---------+----------+-----------------+-----------+
|Country|Population_2020|Yearly_Change|Net_Change|Density_PKm2|Land_Area_Km2|Migrants_net|Ferti
lity_Rate|Median_Age|Urban_Pop_Percent|World_Share|
+-------+---------------+-------------+----------+------------+-------------+------------+-----
---------+----------+-----------------+-----------+
|  India|     1382345085|       0.99 %|  13586631|         464|      2973190|   -532687.0|
2.2|        28|             35 %|    17.70 %|
+-------+---------------+-------------+----------+------------+-------------+------------+-----
---------+----------+-----------------+-----------+
```

License

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.