



Nama : Triyana Dewi Fatmawati
NIM : 2241720206
Kelas : TI – 3D
Nomor : 21
Mata Kuliah : Big Data

Tugas 7 – Spark Docker

Praktikum:

Interaksi dengan Spark di Lingkungan Windows Menggunakan Docker

Pengerjaan:

1. Pull Image Spark Resmi

docker pull apache/spark:latest

```
PS C:\Users\TRİYANA DF> docker pull apache/spark:latest
latest: Pulling from apache/spark
d9802f032d67: Pulling fs layer
3058f73b8f49: Pull complete
f937e0a2086c: Pull complete
0f3083818c14: Pull complete
4d9bb71a5e54: Pull complete
b072aa17899d: Pull complete
5762a181dda2: Pull complete
1ba3910f6ba2: Pull complete
4f4fb700ef54: Pull complete
391ef20df327: Pull complete
Digest: sha256:39321d67b23e2e0953f81b60778f74bf40c40a18dfb0e881e6a38593af60afa1
Status: Downloaded newer image for apache/spark:latest
docker.io/apache/spark:latest
PS C:\Users\TRİYANA DF>
```

2. Menjalankan Spark Master

Sebelumnya buat docker network sebagai berikut

```
PS C:\Users\TRİYANA DF> docker network create spark-net
58decf0eba52c8fe8ac240fd53ba0cbe7b31492a109e402df9e0a9907bc5fbf4
PS C:\Users\TRİYANA DF>
```

Kemudian jalankan spark-master dalam network tersebut

```
PS C:\Users\TRİYANA DF> docker run -d -p 8080:8080 -p 7077:7077 --name spark-master --network spark-net -n 2g --cpus=2 apache
/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.master.Master
b72fc8a231390fb0b122fa55d066cef49fe5fbde0ce4b82173b5925912c88540
```

Kita alokasikan resource untuk memastikan tidak kekurangan resource dalam menjalankan job.

3. Menjalankan Spark Worker

Kita perlu mengalokasikan resource misalnya 2G memori dan 2 core CPU.

```
PS C:\Users\TRIYANA DF> docker run -d --name spark-worker1 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
7ed4dae77c1bb37d69b53862docker run -d --name spark-worker2 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
bc94a69d45c2344c0be1004906ba5156c66d4de8f2b6d6171f27c0d496f559b1
PS C:\Users\TRIYANA DF> docker run -d --name spark-worker3 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
26943cc143d5e97c555843d819cf55a72e2956d5a4b4ed9200bb4b8a057d243d
```

Berikut hasil menggunakan 3 worker.

The screenshot shows the Docker Desktop interface. The 'Containers' tab is active, displaying a list of running containers. Below the list, a terminal window is open, showing the commands used to start the Spark containers.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
spark-master	b72fc9a23139	apache/spark:latest	7077-7077	0.22%	5 minutes ago	[Stop] [Refresh] [Delete]
spark-worker1	7ed4dae77c1b	apache/spark:latest		0.18%	1 minute ago	[Stop] [Refresh] [Delete]
spark-worker2	bc94a69d45c2	apache/spark:latest		0.17%	54 seconds ago	[Stop] [Refresh] [Delete]
spark-worker3	26943cc143d5	apache/spark:latest		0.24%	44 seconds ago	[Stop] [Refresh] [Delete]

Terminal output:

```
g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker
b72fc9a23139f5b0122fa55086c4f9fe5fde0ce4082173b5925912c8854b
PS C:\Users\TRIYANA DF> docker run -d --name spark-worker1 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
7ed4dae77c1bb37d69b53862docker run -d --name spark-worker2 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
bc94a69d45c2344c0be1004906ba5156c66d4de8f2b6d6171f27c0d496f559b1
PS C:\Users\TRIYANA DF> docker run -d --name spark-worker3 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
26943cc143d5e97c555843d819cf55a72e2956d5a4b4ed9200bb4b8a057d243d
PS C:\Users\TRIYANA DF>
```

4. Mengakses Spark Web UI

<http://localhost:8080>

The screenshot shows the Spark Master Web UI at <http://localhost:8080>. The interface displays the Spark Master's status, including the URL, alive workers, cores in use, memory in use, and resources in use. It also shows a table of workers and a section for running and completed applications.

Spark Master at spark://172.18.0.2:7077

URL: spark://172.18.0.2:7077

Alive Workers: 3

Cores in use: 3 Total, 0 Used

Memory in use: 3.0 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20250422085402-172.18.0.3-37801	172.18.0.3:37801	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20250422085424-172.18.0.4-44305	172.18.0.4:44305	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20250422085434-172.18.0.5-44019	172.18.0.5:44019	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

5. Menjalankan Spark Shell

```
Terminal
PS C:\Users\TRIYANA DF> docker run -it --rm --name spark-shell --network spark-net --link spark-master:spark-master apache/spark:latest /opt/spark/bin/spark-shell --master spark://spark-master:7077
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/22 08:57:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://82908dcd13ef:4040
Spark context available as 'sc' (master = spark://spark-master:7077, app id = app-20250422085742-0000).
Spark session available as 'spark'.
Welcome to
  ____              __
 / ___/___  ___  ___/  /___  ___
/ /  / __ \/ __ \/ __/  / __/
/___/_/ /_/_/ /___/_/ /___/_/
version 3.5.5

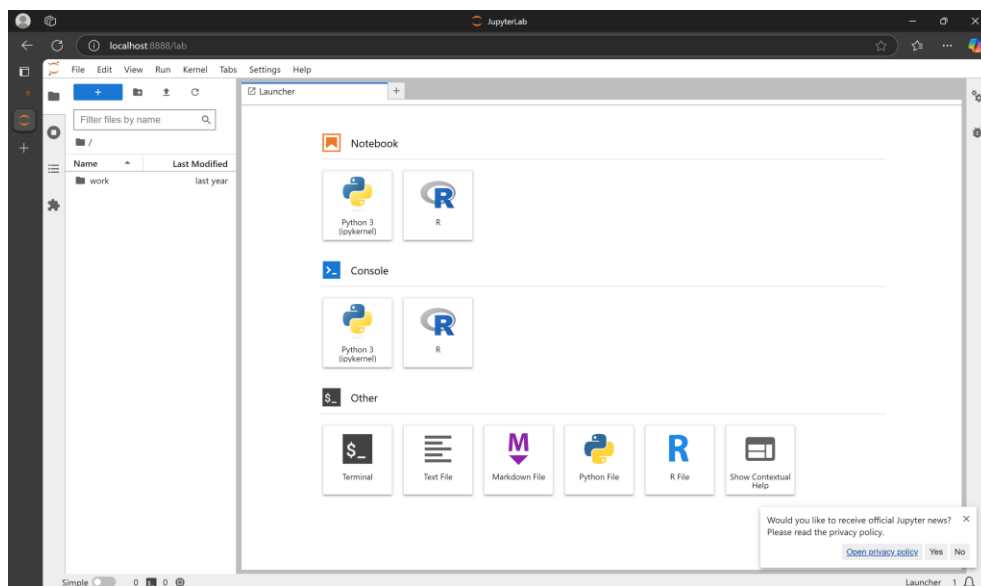
Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.26)
Type in expressions to have them evaluated.
Type :help for more information.

scala> █
```

6. Menggunakan Jupyter Notebook dengan Spark

```
PS C:\Users\TRIYANA DF> docker run -it -p 8888:8888 -p 4040:4040 --network spark-net jupyter/all-spark-notebook
Unable to find image 'jupyter/all-spark-notebook:latest' locally
latest: Pulling from jupyter/all-spark-notebook
```

Setelah itu, akses Jupyter Notebook di: <http://localhost:8888>



7. Untuk menghentikan container:

docker stop spark-master spark-worker

docker rm spark-master spark-worker

```
PS C:\Users\TRIYANA DF> docker stop spark-master spark-worker1 spark-worker2 spark-worker3
spark-master
spark-worker1
spark-worker2
spark-worker3
PS C:\Users\TRIYANA DF> docker rm spark-master spark-worker1 spark-worker2 spark-worker3
spark-master
spark-worker1
spark-worker2
spark-worker3
PS C:\Users\TRIYANA DF> █
```

Contoh Program Word Count dengan Spark di Docker

Berikut adalah contoh program Word Count (menghitung kemunculan kata) menggunakan Apache Spark yang bisa dijalankan di lingkungan Docker:

Cara 1: Menggunakan Spark Shell

1. Jalankan Spark Shell di Docker seperti contoh di atas
2. Ketikkan kode berikut di Spark Shell:

```
scala> val textData = List("Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes Spark easy")
textData: List[String] = List(Hello Spark, Hello Docker, Spark is awesome, Docker makes Spark easy)

scala> val rdd = sc.parallelize(textData)
rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> val wordCounts = rdd
  .flatMap(line => line.split(" ")) // Memecah setiap baris jadi kata
  .map(word => (word, 1))           // Membuat pasangan (kata, 1)
  .reduceByKey(_ + _)              // Menggabungkan jumlah per kata
wordCounts: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> res0: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at flatMap at <console>:24

] = ShuffledRDD[3] at reduceByKey at <console>:24

scala> wordCounts.collect().foreach(println)
[Stage 0:>
Hello Docker
Spark is awesome
Docker makes Spark easy
Hello Spark
```

Untuk keluar dari spark-shell gunakan: `System.exit(0)`

```
scala> System.exit(0)
PS C:\Users\TRİYANA DF>
```

Cara 2: Menggunakan PySpark (Python)

1. Jalankan PySpark Shell di Docker:

Dalam command juga terdapat definisi network juga.

```
PS C:\Users\TRİYANA DF> docker run -it --rm --name pyspark-shell --network spark-net --link spark-master:spark-master apache/spark:latest /opt/spark/bin/pyspark --master spark://spark-master:7077
Python 3.8.10 (default, Feb 4 2025, 15:02:54)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/22 10:22:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to

  / _ \   __ _ _   _   _ \   __ _ _   _
 _/ _ \  / _` | | | | | | \  / _` | | | |
/_/ _ \ /___/ |_| |_| |_| \/_/___/ |_| |_|

version 3.5.5

Using Python version 3.8.10 (default, Feb 4 2025 15:02:54)
Spark context Web UI available at http://45c596fedf0e:4040
Spark context available as 'sc' (master = spark://spark-master:7077, app id = app-20250422102241-0003).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
```

2. Ketikkan kode Python berikut:

```
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.appName("WordCount").getOrCreate()
>>> data = ["Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes Spark easy"]
>>> rdd = spark.sparkContext.parallelize(data)
>>> word_counts = rdd.flatMap(lambda line: line.split(" ")) \
... .map(lambda word: (word, 1)) \
... .reduceByKey(lambda a, b: a + b)
>>> word_counts.collect()
[Stage 0:>
[('Spark', 3), ('awesome', 1), ('Docker', 2), ('easy', 1), ('Hello', 2), ('is', 1), ('makes', 1)]
```

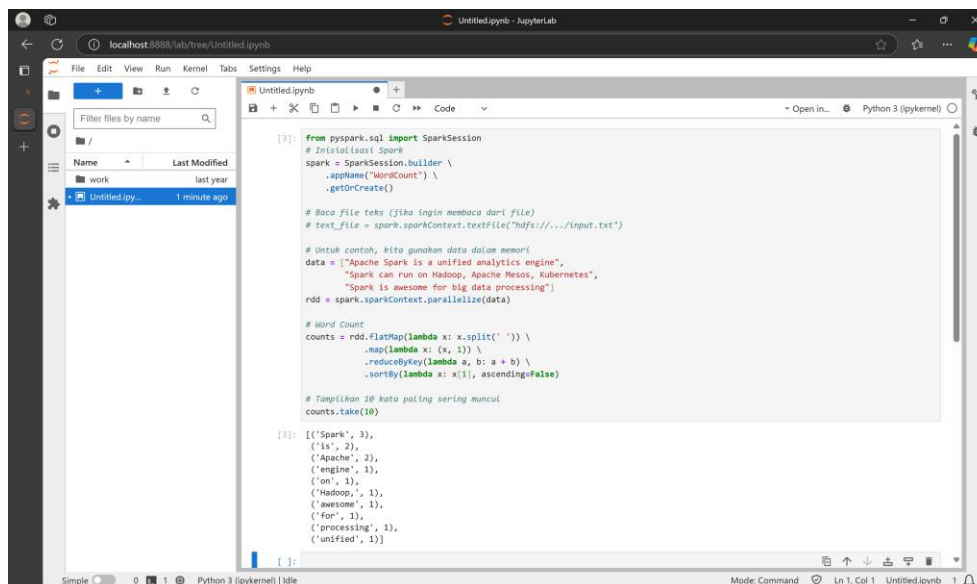
Untuk keluar dari pyspark-shel menggunakan: exit()

```
>>> exit()
PS C:\Users\TRIYANA DF>
```

Cara 3: Menggunakan Jupyter Notebook

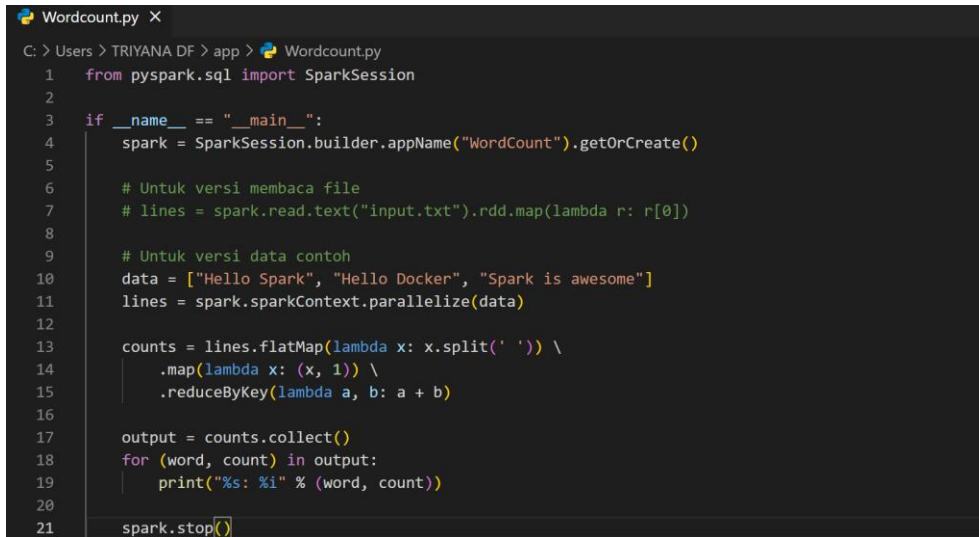
Seperti di container jupyter/all-spark-notebook, akses Jupyter Notebook di:

<http://localhost:8888>



Menjalankan Program sebagai Script

1. Buat file wordcount.py dengan isi berikut:



```
Wordcount.py X
C: > Users > TRIYANA DF > app > Wordcount.py
1  from pyspark.sql import SparkSession
2
3  if __name__ == "__main__":
4      spark = SparkSession.builder.appName("WordCount").getOrCreate()
5
6      # Untuk versi membaca file
7      # lines = spark.read.text("input.txt").rdd.map(lambda r: r[0])
8
9      # Untuk versi data contoh
10     data = ["Hello Spark", "Hello Docker", "Spark is awesome"]
11     lines = spark.sparkContext.parallelize(data)
12
13     counts = lines.flatMap(lambda x: x.split(' ')) \
14                     .map(lambda x: (x, 1)) \
15                     .reduceByKey(lambda a, b: a + b)
16
17     output = counts.collect()
18     for (word, count) in output:
19         print("%s: %i" % (word, count))
20
21     spark.stop()
```

2. Jalankan script, jangan lupa juga mendefinisikan network spark-net

```
PS C:\Users\TRİYANA DF\app> docker run --rm --network spark-net -v ${PWD}:/app --link spark-master:spark-master apache
/spark:latest /opt/spark/bin/spark-submit --master spark://spark-master:7077 /app/Wordcount.py
25/04/22 22:02:28 INFO SparkContext: Running Spark version 3.5.5
25/04/22 22:02:28 INFO SparkContext: OS info Linux, 5.15.167.4-microsoft-standard-WSL2, amd64
25/04/22 22:02:28 INFO SparkContext: Java version 11.0.26
```

Perhatikan dalam command tersebut mendefinisikan akses data ke local.

Program-program di atas akan menghasilkan output seperti:

```
25/04/22 22:12:13 INFO DAGScheduler: Job 0 finished: collect at /app/Wordcount.py:17, took 5.946905 s
Hello: 2
Spark: 2
is: 1
awesome: 1
Docker: 1
25/04/22 22:12:13 INFO SparkContext: SparkContext is stopping with exitCode 0.
```

Tugas:

Menjalankan Apache Spark di Azure

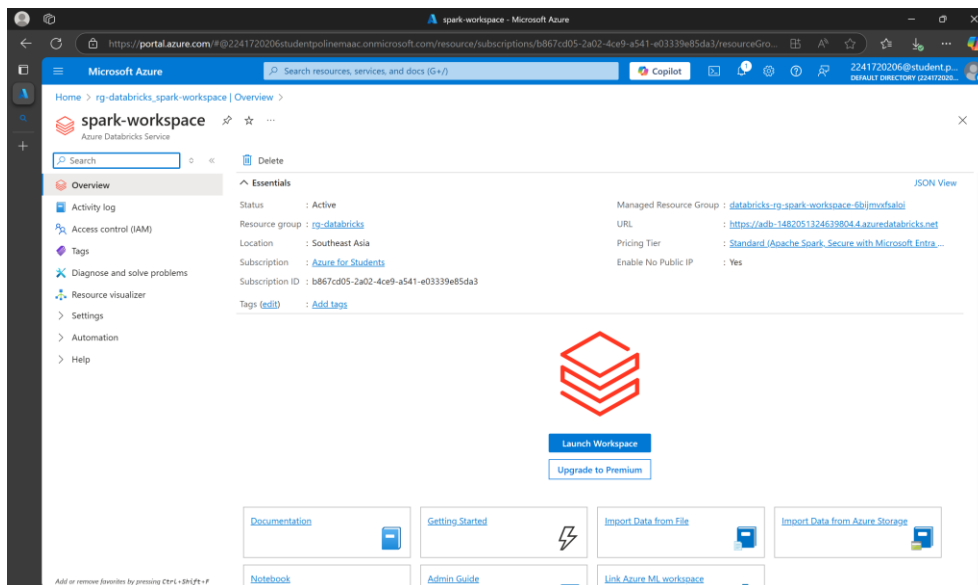
Pengerjaan:

Langkah-langkah untuk menjalankan Apache Spark di Azure:

Disini saya menggunakan Azure Databricks yaitu layanan analytics berbasis Apache Spark yang dikelola sepenuhnya oleh Azure.

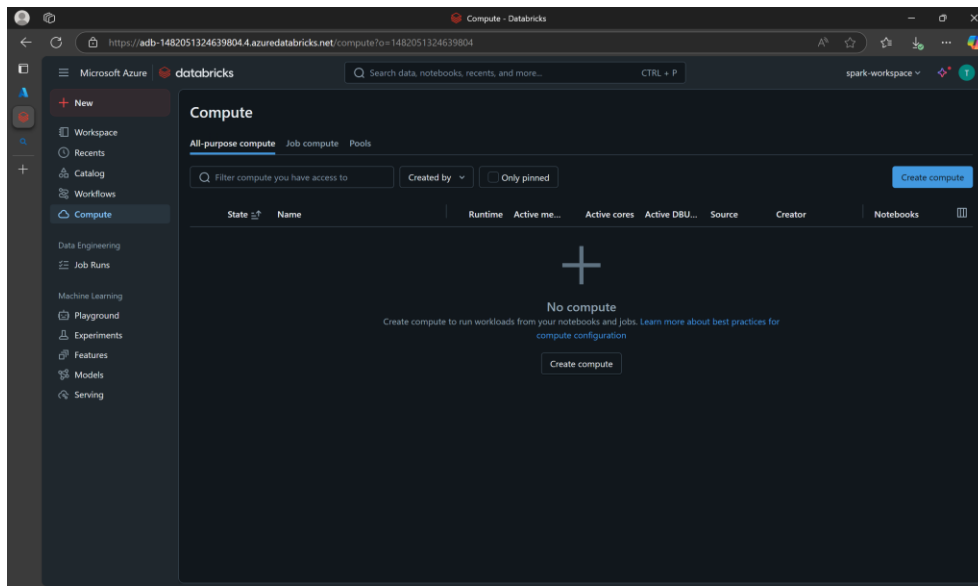
Langkah 1: Buat Azure Databricks Workspace

Pertama, saya membuat layanan Azure Databricks melalui portal Azure untuk menjalankan Spark di cloud.



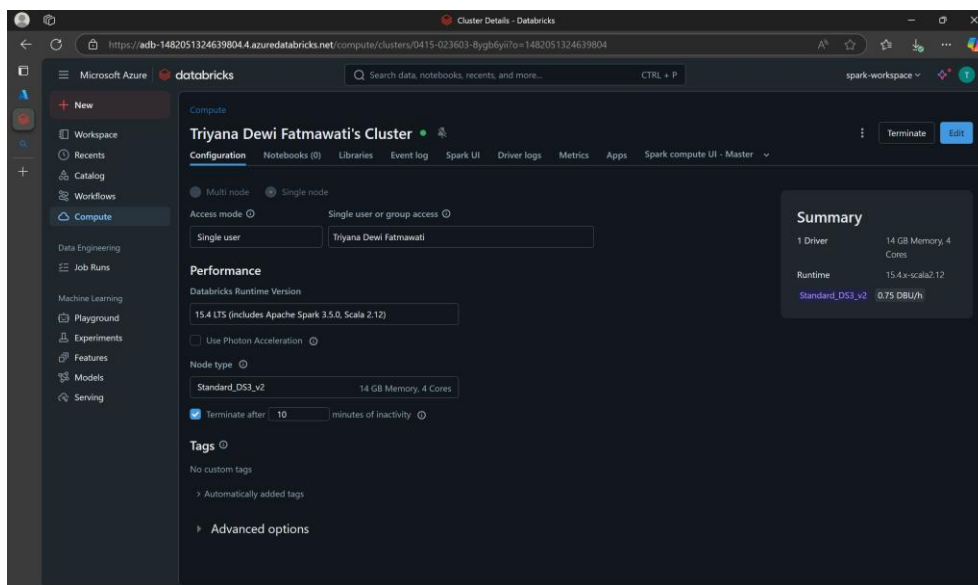
Langkah 2: Masuk ke Databricks Workspace

Setelah workspace berhasil dibuat, saya masuk ke dalamnya dengan klik Launch Workspace untuk mulai menggunakan fitur-fitur yang tersedia.



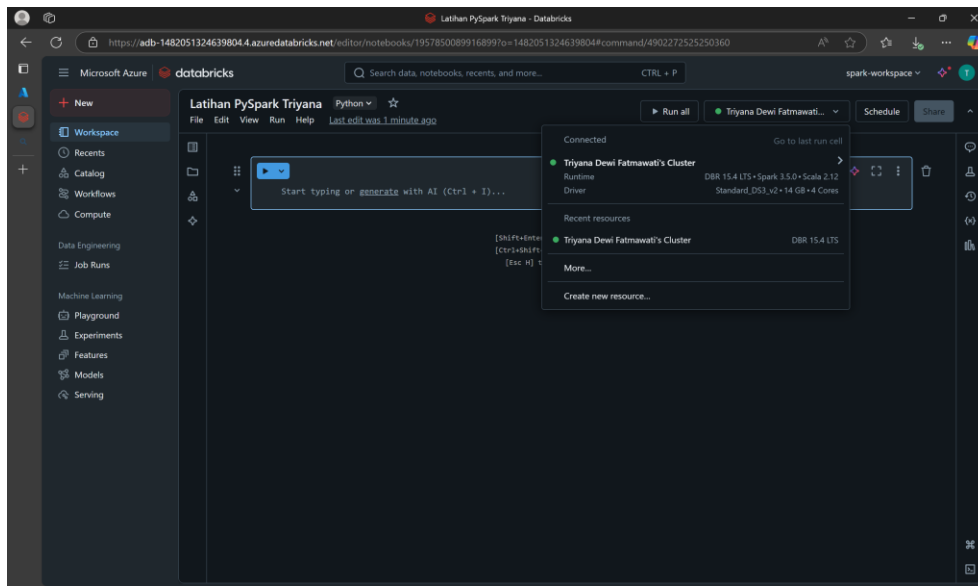
Langkah 3: Buat Cluster

Selanjutnya, saya membuat cluster yang berfungsi sebagai “mesin” untuk menjalankan kode Spark. Saya memilih cluster single node agar lebih hemat dalam penggunaan kredit.



Langkah 4: Buat Notebook & Jalankan Spark

Kemudian, saya membuat sebuah notebook di dalam workspace. Notebook ini digunakan sebagai tempat untuk menulis dan menjalankan kode Spark.



Langkah 5: Tulis Kode PySpark

Terakhir, saya menulis kode PySpark di notebook untuk memastikan bahwa Spark sudah berhasil dijalankan di Azure.

